VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Webové rozhraní IDS systému Snort

Snort IDS Web Interface

2019 Dominik Raška

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Dominik Raška**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Webové rozhraní IDS systému Snort
Snort IDS Web Interface

Jazyk vypracování: čeština

Zásady pro vypracování:

Snort je síťový Intrusion Detection System (IDS) založený na pravidlech. Na základě odposlechu síťové komunikace hledá známé vzorky útoků a v případě nalezení je schopen provádět další akce. Kromě generování protokolu události je schopen zaznamenávat paketová data na disk, kde mohou sloužit k další analýze. Cílem bakalářské práce je vytvořit nástavbu aplikace Snort umožňující přehlednou analýzu logů prostřednictvím webového rozhraní. Inspirací mohou být projekty BASE, OSSIM nebo SnortALog. Vyvinuté rozhraní by mělo být schopné provozu i na méně výkonných routerech typu Alix.

1. Seznamte se s IDS systémy, jejich funkcemi a konfigurací.
2. Proveďte rešerši grafických a webových nástaveb IDS systému Snort.
3. Navrhněte a implementujte vlastní rozhraní umožňující přehlednou vizualizaci zachycených útoků s možností stáhnout zachycené paketové data.
4. Otestujte navržené řešení a výstupy porovnejte alespoň s dvěma jinými systémy umožňujícími vizualizaci protokolů systému Snort.
5. Shrňte dosažené výsledky a popište možnosti dalšího vývoje.

Seznam doporučené odborné literatury:

[1] Chris McNab, Network Security Assessment: Know Your Network, O'Reilly Media; 3 edition, 2016, ISBN 978-1491910955
[2] Rafeeq Rehman, Intrusion Detection With SNORT, Apache, MySQL, PHP, And ACID, Pearson Technology Group, 2007, ISBN 978-0131407336
[3] Diyar Salah Fadhil, Samir Al-Khayatt, Automatic Intrusion Prevention Technique to Improve Network Security, LAP LAMBERT Academic Publishing, 2016, ISBN 978-3659962011

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Mgr. Ing. Michal Krumnikl, Ph.D.**

Datum zadání: 01.09.2017

Datum odevzdání: 30.04.2019

doc. Ing. Jan Platoš, Ph.D.
vedoucí katedry

prof. Ing. Pavel Brandštetter, CSc.
děkan fakulty

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 25. dubna 2019 .....

Na tomto místě bych rád poděkoval hlavně Mgr. Ing. Michalu Krumniklovi, Ph.D. za jeho rady a připomínky, které mi pomohly při této práci, a ještě pomůžou při pracích budoucích.

Abstrakt

Účelem této práce je vytvořit nástavbu aplikace Snort umožňující přehlednou analýzu logů prostřednictvím webového rozhraní, které bude schopné provozu i na routerech s relativně malým výkonem. V úvodu práce je čtenář seznámen s nejznámějšími systémy pro odhalení průniku, jejich funkcemi a konfigurací. Následuje porovnání již existujících rozhraní pro systém Snort a návrh vlastní implementace webové rozhraní. Vlastní implementace je potom řádně zdokumentovaná a otestovaná na více zařízeních.

Klíčová slova: IDS, Snort rozhraní, webové rozhraní, síťové zabezpečení

Abstract

This thesis objective was to implement a Snort extension allowing for convenient analysis of Snort alerts through a web interface. At first, reader is being presented with the theory of intrusion detection systems, how they work and what are some commonly used alternatives. After that, the existing Snort interfaces are being compared while proposing the vision for my own implementation. My application is then thorougly documented and deployed in a couple of testing enviroments.

Key Words: IDS, Snort interface, web interface, network security

Obsah

Seznam použitých zkratek a symbolů

AJAX	–	Asynchronous JavaScript and XML
API	–	Application Programming Interface
ARP	–	Address Resolution Protocol
ASCII	–	American Standard Code For Information Interchange
BSD	–	Berkeley Software Distribution
CPU	–	Central Processing Unit
CSV	–	Comma Separated Values
CUDA	–	Compute Unified Device Architectur
DDoS	–	Distributed Denial Of Service
DNS	–	Domain Name System
GUI	–	Graphical User Interface
HTML	–	Hypertext Markup Language
HTTP	–	HyperText Transfer Protocol
ICMP	–	Internet Control Message Protocol
IDS	–	Intrusion Detection System
IP	–	Internet Protocol
IPS	–	Intrusion Prevention System
JIT	–	Just In Time
JSON	–	JavaScript Object Notation
MD5	–	Message-Digest Algorithm 5
OS	–	Operating System
PDF	–	Portable Document Format
PHP	–	Hypertext Preprocessor
RAM	–	Random Access Memory
SANCP	–	Security Analyst Network Connection Profiler
SCD	–	Self-contained Deployment
SMTP	–	Simple Mail Transfer Protocol
SQL	–	Structured Query Language
SSH	–	Secure Shell
SVG	–	Scalable Vector Graphics
TCL	–	Tool Command Language
TCP	–	Transmission Control Protocol
UDP	–	User Datagram Protocol
VM	–	Virtual Machine
XML	–	Extensible Markup Language

Seznam obrázků

Seznam tabulek

Seznam výpisů zdrojového kódu

1 Úvod

Obor počítačové bezpečnosti přišel paralelně se vznikem informačních technologií. Základní model informační bezpečnosti zmiňuje tři funkce k zajištění ochrany informací – důvěryhodnost, integrita a dostupnost. Po dlouhou dobu byla tato bezpečnost často přehlížena, rizika byla nízká a v tom smyslu bylo učiněno mnoho bezpečnostních kompromisů.

S nástupem internetu na přelomu tisíciletí se ale radikálně změnila situace. Náš svět je dnes propojen jako nikdy dříve. Máme instantní přístup k informacím a počet uživatelů a chytrých zařízení, které komunikují prostřednictvím počítačových sítí, se stále zvyšuje [1]. Stejně tak se ale zvyšují finanční škody způsobené různými síťovými útoky, které poškozují pilíře informační bezpečnosti (DDoS, phishing, malware atd.). Počítačová bezpečnost a speciálně oblast síťového zabezpečení se tak stává velkou výzvou moderní doby.

Jak prohlásil odborník na počítačovou bezpečnost Bruce Schneier:

> *„You can't defend. You can't prevent. The only thing you can do is detect and respond.“*

Velké části slabin v systému zneužitelných k útoku, jinak řečeno zranitelností, se nedá úplně předcházet. Často může dojít k pochybení uživatele vlivem sociálního inženýrství, špatné konfiguraci softwaru apod. Některé typy útoků zase vyžadují jejich detekci v reálném čase a rychlou reakci, aby nedošlo k narušení systému (útoky typu DDoS, útoky hrubou silou). Složitost v této oblasti a neustálý vývoj nových technologií sebou přináší také nové zranitelnosti, které se dají jen těžko předpovídat předem. Efektivním způsobem, jak bojovat s podobnými útoky je použitím systémů pro odhalení průniku, zkráceně IDS (Intrusion detection system). Přestože takový program má své nástroje k odhalování hrozeb, vyžaduje také průběžné intervence ze strany administrátora. V tempu, v jakém se objevují nové útoky, vzniká potřeba je postupně přidávat do zvoleného IDS. Zároveň ne každá hrozba, kterou IDS identifikuje, je nutně hrozbou. Proto je dobré systém vyladit a podle svých potřeb minimalizovat počet falešně pozitivních útoků a nastavit akci proti skutečným útokům. Takový proces často vyžaduje hloubkovou analýzu vzniklého incidentu. Mým úkolem je implementovat webové rozhraní jako nástavbu IDS Snort, která bude sloužit k této analýze potenciálních útoků administrátorem.

V následující kapitole rozeberu obecné funkce a části IDS. Rozdělím je podle oblasti, kterou monitorují a podle způsobu jakým detekují útoky. V druhé části kapitoly se budu věnovat konkrétním implementacím IDS, zejména potom IDS Snort, který je stěžejní pro tuto práci. Dále, v kapitole 3 popíšu významné volně dostupné rozhraní pro IDS Snort, představím jejich architekturu a specifické vlastnosti. Aplikace mezi sebou porovnám, identifikuji některé jejich nedostatky a představím vizi vlastní implementace webového rozhraní. Kapitola 4 dokumentuje vlastní implementaci rozhraní Snort.NET od jejího návrhu až po testování. Nakonec provedu srovnání funkčních požadavků své implementace s ostatními na jejich příkladném praktickém využití síťovým analytikem.

2 Co je to intrusion detection system?

Jako IDS můžeme označit zařízení nebo software, který má za úkol odhalit síťové či systémové aktivity, které představují možné napadení hostitelské sítě či systému, nebo akce, které takovému napadení předcházejí.

Hlavním prvkem softwarových řešení, které lze označit jako IDS, je senzor. Jde o pasivní součást obsahující mechanismy pro detekci podezřelých aktivit na základě dat poskytnutých hostitelským systémem. V závislosti na topologii sítě se určuje umístění případně počet potřebných senzorů na ochranu sítě proti externím, v některých případech i interním hrozbám. Typickým řešením je umístit senzory za každý router, který je na rozhraní chráněné sítě s nedůvěryhodnou externí sítí [2].

IDS systémy lze dále rozdělit podle metody detekce událostí narušující bezpečnost na dva hlavní typy:

1. Podle signatury – Součástí systému je množina pravidel, jenž představuje vzory známých útoků. Pozorovaná data jsou porovnávána s těmito vzory. Takovým způsobem lze dobře určit, o jaký typ útoků se jedná. Efektivita odhalení útoků spočívá ve spektru použitých pravidel a jejich aktuálnosti. Riziko představují tzv. zero-day exploity (bezpečností hrozby, které zatím nejsou obecně známé).

2. Podle anomálie – IDS si stanoví profily představující normální chování uživatele nebo sítě. Anomálií se potom rozumí odchylka od tohoto typického chování, která může být detekována překročením mezních hodnot vybraných atributů (např. počet neúspěšných přihlášení do systému) nebo za pomocí technik umělé inteligence.

Z hlediska dat, se kterými tyto systémy pracují, mluvíme o síťově orientovaných (network intrusion detection system – NIDS) nebo uzlově orientovaných systémech (host intrusion detection system – HIDS).

2.1 NIDS

Monitorují vybraný segment sítě tak, že odposlouchávají síťovou komunikaci, která jím prochází. Pro odposlech je obvykle vybráno jedno nebo více zařízení strategicky tak, aby sledovaly obousměrný tok dat více hostitelů sítě ještě před případným firewallem.

Napadení je nejčastěji identifikováno na základě shody procházejících paketů se známými vzorky útoků, tj. s pravidly. Při takovéto shodě, NIDS zaznamená danou událost v binární nebo textové podobě pro administrátora či další softwarové vrstvy. Možnosti zpracování těchto událostí pomocí webových a grafických aplikací jsou jádrem této práce, kterým se budu věnovat zejména v kapitole 3.

2.2 HIDS

Monitoruje jednotlivé hosty zvlášť a podle více kritérií než jen na základě síťové komunikace. Ve spolupráci s hostitelským operačním systémem analyzují systémová volání, chybová hlášení, logy,

úpravy v souborovém systému a jiné akce. Oproti NIDS, HIDS může navázat na dešifrovaný síťový provoz a odhalit větší množství podezřelých aktivit spolu s odkazem na proces a uživatele, který událost vyvolal. Nevýhodou je rozsah jejich pokrytí, respektive závislost na cílovém hostiteli a operačním systému.

2.3 Intrusion prevention system

V překladu systém prevence proti průniku, dále jen IPS. Funguje jako rozšíření IDS tak, že je postaven do cesty procházejícím paketům a může je zastavit nebo do nich jinak aktivně zasahovat předtím, než je předá dále. Po odhalení útoku IPS umožňuje:

- Zastavit útok tím že:
  - Zahodí paket, případně ještě ukončí příslušné sezení.
  - Zablokuje přístup z IP adresy, účtu, nebo jiného atributu útočníka.
  - Zablokuje veškerý přístup k cílové IP adrese, službě nebo aplikaci.
- Změnit konfiguraci jiných bezpečnostních prvků (typicky firewall).
- Některé IPS dovedou také opravit chyby v kontrolním součtu, defragmentovat a normalizovat pakety nebo odstranit škodlivé části obdržených dat a učinit je tak neškodné. Jednoduchým příkladem budiž odstranění souboru nakaženého virem z přílohy emailu. [3]

IDS může být implementováno jako hardwarové nebo softwarové řešení. Výhodou softwarových IDS je flexibilita jejich konfigurace, jednoduchá údržba a cena, zatímco hardwarové IDS mohou mít daleko větší propustnost [4]. V další podkapitole si představíme nejznámější volně dostupné softwarové implementace IDS s metodou detekce podle signatury – Snort a Suricata. A nakonec také úvod do systémů s metodou detekce podle anomálie – Zeek.

2.4 IDS Snort

Snort[1] je program, jehož hlavní doménou je analýza síťového provozu v reálném čase a spadá do kategorie NIDS s metodou detekce podle signatury.

Původně byl Snort navržen pouze k odposlechu síťové komunikace. Jeho autor, Martin Roesch, chtěl vytvořit lepší alternativu k existujícímu programu Tcpdump pro své vlastní použití. Roesch k odposlechu využil existující knihovnu libpcap, která usnadnila přenositelnost aplikace mezi různými operačními systémy. Data paketů byly zobrazeny v hexadecimální podobě, což Tcpdump v té době neumožňoval. Netrvalo to dlouho a v prosinci 1998 byl Snort zveřejněn. Tehdy obsahoval pouze 1600 řádků kódu ve dvou souborech. Již o měsíc později byl proveden první tah směrem k analýze paketů na základě pravidel a Snort tak mohl být použit jako jednoduchý IDS. Na konci roku 1999 vyšla verze

[1] https://www.snort.org/

1.5, která rozšířila architekturu Snortu do podoby, jak vypadá dnes [5]. Roesch později založil společnost Sourcefire, která si vysloužila ocenění magazínem InfoWorld jako autor jednoho z nejlepších open-source programů všech dob [6]. Následně, v roce 2009 došlo k odkoupení Sourcefire skupinou Cisco Systems.

Za podpory modulárního designu postupně došlo k rozšíření funkčnost Snortu, který dnes dokáže pracovat s velkým počtem protokolů napříč všemi vrstvami TCP/IP modelu a fungovat také na úrovni prevence proti průniku (IPS).

2.4.1 Režimy operace Snortu

Oficiální dokumentace[2] zmiňuje tři hlavní úlohy Snortu, které lze měnit skrze nastavení argumentů příkazového řádku dle výpisu 1.

1. **Sniffer mode**

   Základní režim, ve kterém Snort odposlouchává síťový provoz. Obsah všech TCP/IP paketů převádí do čitelné formy a posílá ho na standardní výstup. Tento režim spustíme pomocí příkazu ve výpisu 1 na řádku 1.

2. **Packet logger mode**

   Obsah paketů je ukládán na disk do adresářové struktury, která odpovídá IP adresám z externí sítě. Alternativně lze zaznamenat všechny pakety do jednoho souboru v jejich binární podobě. Tento formát odpovídá multiplatformnímu standardu pro ukládání paketů knihoven pcap. Viz výpis 1, řádek 2.

3. **NIDS mode**

   Mód pro detekci průniků pomocí analýzy síťového provozu. Snort získané pakety dále defragmentuje a normalizuje, tak aby mohli být vyhodnoceny pomocí nadefinovaných pravidel. Tento režim umožňuje rozsáhlejší nastavení prostřednictvím konfiguračního souboru *snort.conf*, jehož umístění se zadá jako argument -c přepínače. Z pohledu výkonu se nedoporučuje tento režim kombinovat spolu s výstupem na obrazovku. V rámci přístupu k zachyceným útokům máme dvě možnosti nastavení:

   (a) **pasivní přístup** – Funguje pouze jako IDS a do probíhající komunikace nezasahuje. Pro spuštění programu jako démon lze využít přepínač -D. Viz výpis 1, řádek 3.

   (b) **inline přístup** – Umožňuje Snortu aktivní zásah do probíhající komunikace například tím, že odmítne nebo jen zahodí podezřelý paket, který vyhovuje některému drop/reject pravidlu (o pravidlech více v sekci 2.4.4). Jinými slovy jde o rozšíření Snortu o IPS funkce.

   Prakticky to vypadá tak, že Snort vytvoří transparentní most mezi dvěma rozhraními. Jakmile dorazí nějaký paket na jedno z rozhraní, Snort ho porovná se svými pravidly a

[2] https://www.snort.org/documents/snort-users-manual

v případě shody s pravidlem typu drop/reject je paket zahozen. V opačném případě je paket přeposlán na protější rozhraní beze změn. Inline režim aktivujeme přepínačem -Q a zadáním názvů dvojic rozhraní, které takto chceme kontrolovat, podle příkladu ve výpisu 1, na řádku 4.

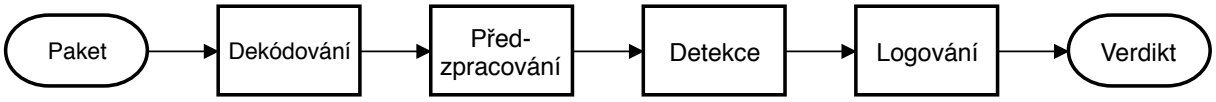
```
1        ./snort -vde
2        ./snort -l ./log -b
3        ./snort -c /etc/snort/snort.conf -D
4        ./snort -c /etc/snort/snort.conf -Q -i eth1:eth2::eth3:eth4
```

Výpis 1: Příkazy ke spuštění Snortu

2.4.2 Konfigurace Snortu

Konfigurační soubor snort.conf je logicky rozdělen do 6 částí. Přičemž části 2.–5. odpovídají sledu procesů znázorněných na obrázku 1.

Obrázek 1: Postup zpracování dat Snortem v režimu NIDS



1. **Nastavení proměnných**

   Obsahuje deklaraci proměnných, které můžou být požity v dalších částech souboru a v pravidlech. Nezbytné je správné nastavení proměnné HOME_NET, respektive adresy chráněné interní sítě.

   ```
   ipvar HOME_NET 192.168.0.0/24
   ```

   Následuje specifikace dostupných serverů a portů, na kterých operují. A nakonec cesta k pravidlům užívaná v poslední části tohoto souboru.

2. **Nastavení dekodéru**

   Dekodér je jeden z prvních procesů Snortu při práci se zachyceným paketem. Jeho úkol spočívá v determinaci použitých komunikačních protokolů a zaznamenání údajů z jejich hlaviček dohromady s umístěním a velikostí datového pole.

   Během tohoto bezstavového procesu může dekodér odhalit chyby a anomálie související s obsahem hlaviček specifických protokolů. Například, mějme ethernetový rámec, který zapouzdřuje IPv4 paket o velikosti menší než 20 bajtů. Tak malá velikost je anomální a neslučitelná s definicí IPv4 protokolu.

   Snort zde umožňuje aktivaci upozornění na tyto problémy, popřípadě úplné zahození postižených paketů v režimu inline.

3. **Nastavení preprocesorů**

   Preprocesory umožňují rozšíření Snortu o funkce využívající stav probíhající komunikace. Typicky jsou aktivovány preprocesory zajišťující sestavování IP fragmentů nebo TCP segmentů do podoby původní PDU (protokolová datová jednotka). Dále máme možnost tyto data znormalizovat a odhalit anomálie a útoky spojené s protokolem dané PDU. Takto se lze bránit proti celé řadě exploitů, které zneužívají slabin ARP (ARP spoofing), SSH, DNS, HTTP a dalších protokolů převážně aplikační vrstvy TCP/IP modelu.

4. **Nastavení detekčního enginu**

   V této části lze nastavit limity a optimalizaci pro porovnávání obsahu paketů s regulárními výrazy v pravidlech. Cílem je minimalizovat a vyvážit potřebný čas a paměť během odhalování útoků.

5. **Nastavení výstupních pluginů**

   Definujeme typy výstupních souborů popsaných v sekci 2.4.3.

6. **Volba pravidel**

   V poslední části je nutné uvést soubory s pravidly, která budou použitá během procesu detekce útoků.

```
include $RULE_PATH/icmp.rules
```

2.4.3 Výstupní pluginy

V případě úspěšné detekce útoku Snort vygeneruje zprávu popisující danou událost, tzv. alert. Administrátor má možnost zvolit jeden nebo více formátů, v jakém se událost uloží:

- **Full alert**

  Ukládá alerty do souboru. Obsahuje signaturu popisující, k jakému útoku došlo a prioritu, která se k ní vztahuje. Dále přesný čas zaznamenání události, zdrojovou i cílovou IP adresu a další data z hlavičky paketu.

```
[**] [1:528:3] BAD TRAFFIC loopback traffic [**]
[Classification: Potentially Bad Traffic] [Priority: 2]
09/23-09:09:22.581891 10.29.12.177:161 -> 127.0.0.1:162
UDP TTL:57 TOS:0x0 ID:40247 IpLen:20 DgmLen:211 Len: 191
[Xref => http://rr.sans.org/firewall/egress.php]
```

- **Fast alert**

  Ukládá alerty do souboru ve zredukovaném formátu.

```
01/31-17:37:40.008521 [**] [1:221:1] DDOS TFN Probe [**]
[Classification: Attempted_Information_Leak] [Priority: 2]
{ICMP} 1.2.3.4 -> 192.168.1.97
```

- **Syslog**

  Snort zasílá logy do systémového logovacího zařízení. Uživatel má možnost vybrat soubor a úroveň závažnosti zasílaných logů. K ukládání alertů zvolíme závažnost `LOG_ALERT`.

- **Unixsock alert**

  Snort vytvoří unixový soket, do kterého zasílá alerty a data příslušných paketů.

- **CSV alert**

  Ukládá alerty do CSV souboru. Obsažená data lze přizpůsobit požadavkům administrátora.

- **Unified2**

  Data jsou uložená do souboru v binární podobě. Snort konfigurace nabízí 3 možnosti struktury unified2 výstupu:

  - log_unified2 – obsahuje kopii paketu, který vyvolal danou událost.
  - alert_unified2 – obsahuje alert (referenci na typ události, čas události, prioritu, zdroj a cíl útoku).
  - unified2 – obsahuje alert i paket který jej vyvolal.

- **Pcap**

  Ukládá podezřelé pakety ve standardním formátu knihoven pcap, který je vhodný pro dodatečnou analýzu zachyceného síťového provozu. Mezi populární paketové analyzátory, jenž s tímto formátem pracují, patří například textový nástroj tcpdump nebo program Wireshark.

2.4.4 Snort pravidla

Jedná se o textový popis vybraných atributů paketu nebo sekvence paketů, která představuje potenciální narušení informační bezpečnosti. Balík pravidel je dostupný ke stažení z oficiálních webových stránek Snortu[3]. Talos je dedikovaná skupina bezpečnostních expertů ze Sourcefire, která tyto pravidla průběžně aktualizuje. Archiv je rozdělen do více souborů s pravidly pojmenované podle druhu útoku (např. „ddos.rules“). Jednotlivá pravidla jsou rozdělena do dvou částí:

[3] https://www.snort.org/downloads#rules

1. **Rule header**

   Záhlaví pravidla. Určuje komunikační protokol (IP, TCP, UDP, ICMP), cílovou a zdrojovou IP adresu/podsíť, cílový a zdrojový port (relevantní pouze pro TCP/UDP protokol) a akci, která se má provést. Snort nativně podporuje tyto akce:

   - alert – Generuje alert podle vybraného výstupního pluginu a uloží paket, který tento alert vyvolal
   - log – Uloží pouze paket.
   - sdrop – Zablokuje paket (vyžaduje režim inline).
   - drop – Zablokuje a uloží paket (vyžaduje režim inline).
   - reject – Zablokuje a uloží paket. V případě, že se jedná o protokol TCP, odešle TCP reset. V případě, že jde o protokol UDP, odešle odpověď hlásící nedostupnost cílového portu (vyžaduje režim inline).
   - pass – Ignoruje paket.

2. **Rule options**

   Uvádí, o jaký útok se jedná pomocí obecných parametrů:

   - msg – Popis útoku. Využívá ho výstupní engine při generování alertu.
   - reference – Umožňuje přiřadit odkaz na externí zdroj s podrobnějšími informacemi k danému útoku.
   - sid – Používá se k unikátní identifikaci daného pravidla, respektive útoku. Hodnoty menší než 100 jsou vyhrazeny do budoucnosti. Hodnoty do 999 999 jsou určeny pro oficiální distribuci pravidel Snortu. Vyšší hodnoty lze využít k definici lokálních pravidel.
   - rev – Určuje revizi, respektive verzi pravidla.
   - gid – Identifikuje, jaký proces tento útok vyvolá. Výchozí hodnota je 1, která je spojená s obecným detekčním enginem.
   - classtype – Snort poskytuje několik obecných kategorií pro útoky definovaných v souboru classification.config, který je součástí oficiální distribuce pravidel. Každá kategorie má svůj popis a výchozí prioritu pro příslušné pravidla.
   - priority – Určuje závažnost pravidla. Zpravidla jde o hodnoty 1–4 (kde nižší hodnota značí vyšší závažnost).

   Následuje velký počet volitelných parametru umožňující identifikovat útok pomocí vzorů v datovém poli nebo v hlavičkách protokolů. Zvláštním parametrem je tzv. flowbits, který ve spolupráci se Session preprocesorem dokáže určit stav TCP sezení a stav komunikace na úrovni aplikačního protokolu.

   Posledním krokem při detekci útoku je evaluace parametru detection_filter, kterým lze stanovit maximální počet shod předtím, než Snort vygeneruje oznámení o útoku.

Dovolím si uvést příklad pravidla ve výpisu 2, které vygeneruje alert po 30 neúspěšných přihlášeních v rozmezí 60 sekund na SSH server (situovaný na IP adrese 192.168.0.104).

```
alert tcp any any -> 192.168.0.104 22 (msg:"SSH Brute Force Attempt"; flow:
    established,to_server; content:"SSH"; nocase; offset:0; depth:4;
    detection_filter:track by_src, count 30, seconds 60; sid:1000001; rev:1;)
```

Výpis 2: Příklad pravidla pro IDS Snort

2.4.5 Snort 3.0

V polovině roku 2017 Cisco vydalo první alfa verzi projektu pod názvem Snort++. Jde o novou generaci Snortu s modernějším designem a novými funkcemi.

Došlo ke značnému zlepšení výkonu při inspekci paketů za použití více vláken. Konfigurace byla zjednodušena s Lua JIT kompilátorem. Zároveň byla vylepšena podpora pro operační systém Windows. Změny proběhly také v parametrech pravidel, což učinilo pravidla pro Snort 3.0 zpětně nekompatibilní. [7]

2.5 IDS Suricata

Suricata[4] je další open-source NIDS vydaný v roce 2010 neziskovou organizací pod názvem Open Information Security Foundation (OISF). Suricata v mnoha ohledech napodobuje filozofii Snortu a snaží se vylepšit některé jeho nedostatky – zejména v oblasti výkonosti. S narůstajícími počty pravidel a objemu dat protékající sítí se jednovláknové paradigma Snortu stává nedostačující. Suricata proto pracuje za běhu více vláken pro adekvátní využití potenciálu vícejádrových procesorů. Zároveň podporuje paralelní zpracování na grafických procesorech s architekturou CUDA pro detekci shod se vzory v pravidlech.

Suricata si udržuje kompatibilitu s pravidly pro Snort. K zajištění ochrany proti více druhům hrozeb je vhodné použít také doplňující balík pravidel Emerging Threats Ruleset [8].

Podporované výstupní formáty zahrnují textové alerty událostí, DNS log, HTTP log, unified2, pcap a JSON. Výhodou výstupního formátu JSON je přirozená integrace s populárními nástroji pro správu logů jako je Logstash.

2.5.1 Extrahování souborů

Jednou z funkcí, která obohacuje Suricatu, je schopnost extrahovat soubory z HTTP/SMTP streamu. V rámci pravidel aktivujeme ukládání souborů pomocí klíčového slova *filestore*. Dalšími parametry můžeme také filtrovat soubory pomocí jména, koncovky, velikosti nebo MD5 otisků.

Následující pravidlo vygeneruje alert a uloží všechny PDF soubory, jejichž MD5 otisk se shoduje s některým otiskem v souboru md5-blacklist.

[4] https://suricata-ids.org/

```
alert http any any -> 192.168.0.104 any (msg:"Nalezen zakázaný PDF soubor";
    fileext:"pdf"; filemd5:md5-blacklist; filestore; sid:2; rev:1;)
```

Výpis 3: Příklad pravidla pro IDS Suricata

2.5.2 Automatická detekce aplikace

Suricata rozšířila logiku pravidel o detektory aplikační vrstvy. Umožňuje tak snadno identifikovat kupříkladu HTTP komunikaci nezávisle na komunikačním portu tímto způsobem:

```
alert http $HOME_NET -> $EXTERNAL_NET any ...
```

Výpis 4: Příklad pravidla pro IDS Suricata demonstrující detekci protokolů aplikační vrstvy

V momentě, kdy Suricata detekuje HTTP komunikaci, naplní také své HTTP buffery, na které se lze odkazovat v pravidlech. Tyto buffery slouží pro vyhledávání obsahu v cílených místech HTTP zprávy. Pravidlo v dalším příkladě filtruje pouze HTTP požadavky s metodou GET za použití bufferu *http_method*.

```
alert http $HOME_NET -> $EXTERNAL_NET any {content:"GET"; http_method; ...}
```

Výpis 5: Příklad pravidla pro IDS Suricata demonstrující využití bufferů aplikační vrstvy

2.6 Zeek

Zeek[5], původně známý pod názvem Bro, je všestranný síťový analyzátor a zároveň pasivní NIDS. Jeho počátky sahají do roku 1995, nicméně přelomového růstu se dočkal až po americkém státním grantu v roce 2010. Je šířen pod svobodnou licencí BSD.

Jeho architektura se skládá z dvou hlavních komponent. Event engine sbírá pakety v online či offline režimu a spouští sérii událostí. Tyto události jsou obslouženy uvnitř Zeek scriptů, které vykonává komponenta *script interpreter*. Unikátnost a síla Zeeku spočívá v těchto skriptech napsaných v turingovsky ekvivalentním jazyce strukturálně podobnému C++. Uvnitř skriptů lze provádět stavovou analýzu síťového provozu včetně automatické detekce aplikační vrstvy nebo extrahování souborů a administrátor má plnou kontrolu nad tím, jaké informace se následně uloží do logů. Zeek ve svém výchozím nastavení ukládá základní informace každé aktivity v síti. Taková data jsou užitečná k detekci anomálií nebo investigaci ucelené komunikace po vzniklém incidentu.

Standardem na poli NIDS však zůstává Snort a jeho klony kvůli svým volně dostupným pravidlům, které zajišťují kvalitní ochranu proti známým hrozbám [9]. Zeek je vhodný pro doplnění klasického NIDS, kde jeho účel spočívá v hledání anomálií a nových hrozeb.

[5] https://www.zeek.org/

3 Rešerše webových a grafických rozhraní pro IDS Snort

V této části se budu věnovat rozboru dostupných aplikací k vizualizaci hrozeb odhalených pomocí IDS Snort. Jejich primárním úkolem je usnadnit práci síťových analytiků tím, že prezentují Snort alerty v přehledné formě doplněné o užiteční informace, které s útokem souvisejí. Často také obsahují funkce pro vyhledávání útoků podle jejich parametrů (zdrojové IP adresy, závažnosti, protokolu apod.) nebo grafické elementy znázorňující statistická data z více útoků najednou.

3.1 SnortAlog

Jedná se o skript v jazyce Perl, který pracuje přímo s textovými logy Snortu ve formátu Fast alert, Full alert nebo Syslog[6]. Po spuštění tyto útoky sumarizuje a vygeneruje hlášení ve zvoleném výstupním formátu. Na výběr máme ASCII, HTML nebo PDF formát. Výsledkem jsou různá statistická data ve formě tabulek, která můžou být vizualizována formou grafů. Tyto data představují četnost útoků podle času útoku, zdrojové/cílové IP adresy, protokolu, cílového portu, metody apod. Výsledek lze dále také doplnit o překlad IP adres na doménová jména a stát původu. Vstupní data mohou být filtrována za použití argumentů popsaných v technické dokumentaci programu.

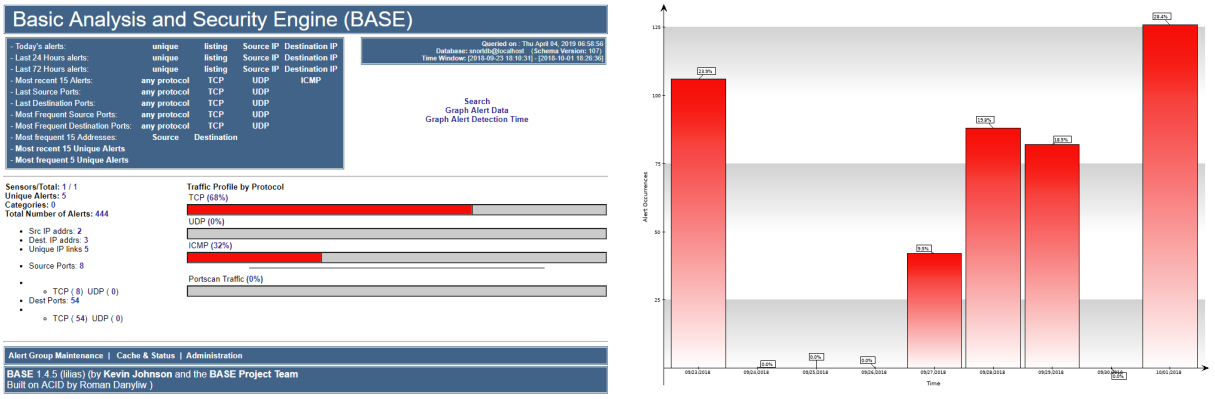
SnortAlog funguje na principu On-demand, který zaručuje jeho nízkou hardwarovou náročnost. Ke své funkci navíc nevyžaduje žádnou databázi, což přispívá k jednoduchému použití a konfiguraci. Přesto, jde pouze o zevrubné hlášení, které neobsahuje nástroje k podrobné investigaci a analýze zachycených útoků.

Existuje řada programů pracující na stejném principu jako SnortAlog (například SnortSnarf[7]). Výsledná data a možnosti jejich zpracování však podle mého bádání tvoří jen podmnožinu toho, co nabízí SnortAlog.

3.2 BASE

BASE[8] (Basic Analysis and Security Engine) je nástroj, určený k vizualizaci a analýze výstupních dat z IDS Snort prostřednictvím webového rozhraní. Jeho design, postavený na jazyku PHP, je odvozen od projektu ACID (Analaysis Console for Intrusion Databases), který vznikl již v roce 2001. Pro svou funkčnost vyžaduje Snort alerty nahrané do některé relační databáze (MySQL, PostgreSQL, Microsoft SQL Server, Oracle Database Server). Snort byl během svého vývoje odlehčen o plugin umožňující výstup do relačních databází. Tuto mezeru je nutné vyplnit některou third-party komponentou. Nejčastěji se využívá spooler Barnyard[9], který je schopen v reálném čase interpretovat unified2 výstup Snortu a ukládat data do vybrané relační databáze. Entitně vztahový model výstupní databáze byl převzatý z odstraněného výstupního Snort pluginu, což zaručuje zachování kompatibility také

[6] http://jeremy.chartier.free.fr/snortalog/
[7] https://sourceforge.net/projects/snortsnarf/
[8] https://sourceforge.net/projects/secureideas/
[9] https://github.com/firnsy/barnyard2



(a) Úvodní stránka

(b) Graf počtu útoků podle dnů

Obrázek 2: BASE interface

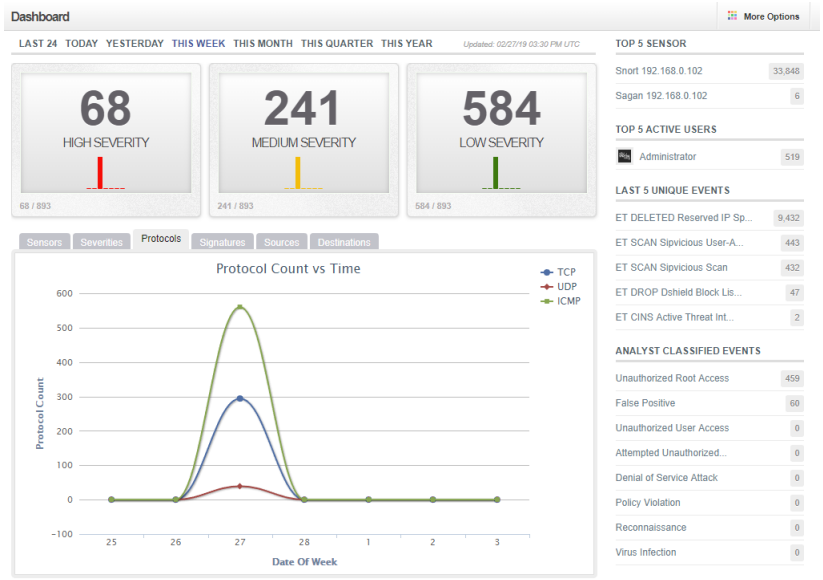
s BASE. Nevýhodou tohoto modelu a zároveň tedy i rozhraní BASE je absence podpory moderního protokolu IPv6.

Samotné rozhraní nabízí přehledovou stránku obsahující souhrnné informace všech zaznamenaných útoků a jejich zastoupení podle použitého protokolu dle obrázku 2. Uložené útoky lze dále filtrovat podle řady parametrů a zobrazit obsah paketů, které tento útok vyvolaly. Informace obsažené v paketu jsou logicky rozdělené na úrovni protokolů TCP/IP modelu. Datové pole paketů je prezentováno v hexadecimální i textové podobě a lze ho stáhnout jako binární soubor. Celý paket může být také exportován do formátu pcap k dodatečné analýze vzniklého incidentu. Pod podmínkou instalace správných PHP rozšíření, zvládá BASE také vykreslit řadu grafů v jedné ze tří forem (sloupcové, spojnicové, koláčové). Rozhraní dále nabízí několik odkazů k Whois registrům, tj. k informacím, které jsou spojené se zdrojovou IP adresou zachycených útoků. Možností je také geolokace dané IP adresy prostřednictvím offline modulu.

3.3 Snorby

Snorby[10] je webová aplikace fungující na platformě Ruby on Rails. K dispozici je několik linuxových distribucí obsahující Snorby, který monitoruje výstup z více IDS najednou (Snort, Suricata a Sagan) podle zvolené konfigurace. Vzhledem k tomu, že se jedná o mladší projekt, má v porovnání s BASE modernější design (viz obrázek 3) a funkce vyplývající z nástupu Web 2.0. Administrátor má možnost si jednotlivé události označit, roztřídit je do kategorií nebo k nim přidat svou poznámku. Ukládat lze také vyhledávací řetězce, respektive filtry pro události. Detailní informace o útoku si uživatel smí exportovat do formátu XML nebo zaslat na e-mail. V základní konfiguraci však schází možnost stáhnout paket ve formátu pcap. Náhradou pro tuto funkci je integrace a interface pro OpenFPC[11].

[10] https://github.com/Snorby/snorby
[11] http://www.openfpc.org/



Obrázek 3: Snorby interface

3.3.1 Záznam paketů s OpenFPC

Silnou zbraní na poli síťové bezpečnosti je systém pro zaznamenávání veškeré komunikace. Richard Bejtlich uvádí: „*Full content data is the most flexible form of network-based information. It is a rich form of evidence offering detail and opportunities seldom found elsewhere.*“ [10]

V praxi to znamená, že analytik má možnost spojit vzniklou událost, na kterou Snort upozornil, se záznamem celé komunikace, respektive síťové relace, která tento incident vyvolala. Překážkou této funkce je její hardwarová náročnost. Ve většině případů je proto záznam omezen pouze na určitý protokol nebo IP adresu. Mimo jiné samotná takto uložená data mohou být terčem útoku, a představovat tak bezpečnostní riziko jejich odcizení. OpenFPC je jeden z open-source pluginů, který výše zmíněné implementuje.

3.4 Sguil

Jedná se open-source aplikaci pro komplexní monitoring sítě[12]. Sguil spojuje výstupy z vícero nástrojů pro síťovou bezpečnost do jednoho rozhraní, tak aby maximálně usnadnil práci síťových analytiků. Také používá svou vlastní databázi pro většinu získaných dat, což umožňuje používat vyhledávací SQL dotazy přímo z jeho GUI.

Rozhraní splňuje funkce svých webových alternativ, ale nejde jen o klasický prohledávač událostí. Sguil zaujímá filozofii událostmi řízené analýzy, přičemž vyžaduje nějakou reakci na každou vzniklou událost. Analytik má za úkol kategorizovat vzniklý incident, případně ho předat dále senior analytikovi.

[12] https://bammv.github.io/sguil/index.html

3.4.1 Architektura

Podle modelu, který je vidět na obrázku 4, je systém rozdělen do tří vrstev:

- **Sguil senzory**

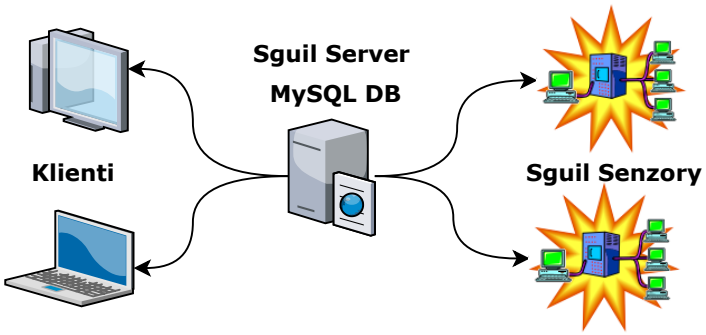
  Každý senzor monitoruje jeden síťový uzel, na kterém sbírá různá data. V systému jsou spuštěné dvě instance Snortu, přičemž jedna instance sbírá alerty v režimu NIDS a druhá je spuštěná v režimu packet logger. Kromě toho Sguil využívá dvou nástrojů, které extrahují data z jednotlivých sezení – jde o SANCP (viz sekce 3.4.2) a Tcpflow (viz sekce 3.4.3).

- **Sguil Server**

  Na serveru je spuštěn Sguil daemon, který přijímá data ze všech senzorů a obsluhuje dotazy od klientů. Snort alerty a data sezení zapisuje ihned do své MySQL databáze. Paket logy zůstávají uložené na senzorech a při potřebě jsou odeslány na server.

- **Klienti**

  Prezentační vrstvu tvoří aplikace napsaná v platformě nezávislém jazyce Tcl.



Obrázek 4: Sguil: 3-vrstvá architektura

3.4.2 SANCP

Jedním ze vstupů jsou data pořízená programem SANCP[13], který agreguje informace spjaté s TCP/IP relacemi. Patří zde IP adresy a porty obou účastníků komunikace, počet přenesených paketů, počet přenesených bytů, čas začátku a konce relace. Data zahrnují komunikaci skrze protokoly TCP, ICMP i UDP.

3.4.3 Tcpflow

Tcpflow[14] sestavuje proud paketů probíhající TCP komunikace do správného pořadí a ukládá přepis přenesených dat v ASCII formě do souborů. Sguil používá Tcpflow k vytvoření výpisu zpráv na úrovni

[13] https://sourceforge.net/projects/sancp/
[14] https://github.com/simsong/tcpflow

aplikační vrstvy ze záznamu paketů. Dokáže tak spojit Snort alert s příslušnou TCP relací a zobrazit zprávy odeslané v obou směrech tohoto spojení.

3.5 Shrnutí

Nyní se pokusím vyjmenovat signifikantní funkce a možnosti, které některé rozhraní mají a tímto způsobem představit vizi mé implementace, její funkční požadavky a zamýšlený způsob integrace těchto funkcí cílený na dobrou použitelnost.

Stručný přehled vybraných parametrů všech rozhraní popsaných v rámci rešerše jsou k nahlédnutí v tabulce 1. Nutno dodat, že popis se vztahuje k aktuálním stabilním verzím těchto rozhraní, viz tabulka 1.

Platforma – Zvolená platforma případně jiné závislosti můžou omezit možnost nasazení aplikace pouze na konkrétní druh operačního systému. Přestože Snort i část diskutovaných rozhraní je v tomhle ohledu plně multiplatformní, mou implementaci budu cílit na systém Linux, který je v praxi nejpoužívanější.

Datová základna – Určuje to, s jakými daty rozhraní pracuje a je schopné prezentovat uživateli. Na první pohled by se dalo říct, že čím více dat, tím lepší a detailnější analýza zaznamenaných hrozeb. Na druhou stranu, jak už bylo zmíněno v sekci 3.3.1, je potřeba vzít úvahu výkon a kapacitu použitelných serverů a provést kompromis. Moje práce je cílená na méně výkonné zařízení, a proto se vyhnu ukládání záznamu veškeré komunikace.

Vyhledávání – Jde o možnost filtrovat Snort alerty podle jejich základních parametrů. Většina aplikací tuto funkci zvládá dobře a není důvod ji vynechat. Užitečným zlepšením však může být uložení tohoto filtru pomocí HTTP cookie.

Vizualizace statistik – Statistická data zobrazující například nejčastější zdroj, cíl, typ nebo protokol použitý k útoku pomocí různých grafů jsou standardem napříč rozhraními a mé řešení nebude výjimkou.

Propojení komponent – Určuje, jak dobře jsou informace na webu propojeny. Jinak řečeno, jak rychle je uživatel schopen přecházet mezi komponenty, které spolu souvisejí. SnortAlog takové propojení nemá a obsahuje výhradně statistická data. Sguil má propojení v pořádku. Komponenty uvnitř BASE a Snorby mají dobré propojení s výjimkou grafů. Propojení těchto grafů s filtrem alertů představuje prostor pro zlepšení, kterého využiji ve své práci.

Možnosti pro export – Nejčastěji se objevuje možnost exportovat celý paket nebo sérii paketů z podezřelé komunikace do formátu pcap. Funkce má své jednoznačné opodstatnění při detailní analýze útoku s využitím specializovaných nástrojů jako je Wireshark[15]. Ve své aplikaci půjdu o krok dál a pokusím se přidat jednoduchou komponentu, která napodobí rozhraní programu Wireshark pomocí HTML tabulky.

Geolokace IP – Kontaktní informace o IP adresách jsou obyčejně získávána z online Whois registrů, kde z pole „country“ lze zjistit stát, kterému byl daný rozsah IP adres přiřazen. Dostupnost

[15] https://www.wireshark.org

takových služeb je relativně spolehlivá a data jsou aktuální. Nevýhodou ale může být rychlost získávání těchto údajů, když je potřeba vyřešit mnoho IP adres naráz. Alternativní možností je průběžně, typicky jednou za měsíc, aktualizovaná offline databáze.

Podpora IPv6 – I přesto, že Snort má již plnou podporu pro nejnovější internetový protokol, velká část webových rozhraní zatím zaostává. Více než o samotném protokolu to vypovídá něco o stáří a vývoji těchto aplikací. Poslední úpravy SnortAlogu nebo BASE jsou starší více než 8 let. Snorby sice IPv6 ovládá, ale je potřeba následovat velmi nestandardní postup při instalaci. Sguil IPv6 útoky zaznamenává bez zdrojové či cílové IP adresy.

Hardwarové nároky – Požadavky na hardware v tomto případě souvisejí hlavně s počtem dat, jež jsou sbírána a posléze analyzována pro prezentaci. Tento proces probíhá nejlépe v reálném čase a s co nejmenším zpožděním. Svou aplikaci hodlám optimalizovat a otestovat míru dlouhodobého využití prostředků, tak aby byla schopna plynulého provozu na dnešních routerech jako jsou modely řady apu2 od PC engines[16].

[16] https://www.pcengines.ch/apu2.htm

Tabulka 1: Přehled rozhraní pro IDS Snort a jejich funkcí

	Platforma	**Datová základna**	**Vyhledávání alertů**	**Vizualizace statistik**	**Propojení komponent**	**Export**	**Geolokace**	**IPv6**	**HW nároky**
SnortAlog 2.4.3	Perl	pouze alerty	ne	tabulky s grafy	žádné	ne	Whois	ne	velmi nízké
BASE 1.4.5	PHP 5.6	alerty s pakety	ano	grafy	dobré	pcap	Whois	ne	nízké
Snorby 2.6.2	Ruby on Rails	alerty s pakety úplný záznam	ano	grafy	dobré	pcap	GeoLite	ano	střední
Sguil 0.9.0	Tcl	alerty s pakety data sezení úplný záznam	ano	ne	výborné	pcap	Whois	ne	vysoké

4 Vlastní implementace webového rozhraní - Snort.NET

Cílem této práce bylo vytvořit vlastní webové rozhraní pro přehlednou vizualizace logů IDS Snort. S ohledem k možnému nasazení i na méně výkonných routerech jsem při implementaci kladl důraz na nízkou hardwarovou náročnost a minimum závislostí. Zároveň jsem využil výhod moderního webového frameworku a použil nejnovější webové technologie pro grafickou vizualizaci statistických dat.

4.1 Návrh

Funkční požadavky byly popsány rámci kapitoly 3.5. V této části představím návrh pro některé technické detaily, které z těchto požadavků často přímo vyplývají.

4.1.1 SW platforma

Aplikace je postavená na platformě .NET Core a přidruženém webovém frameworku ASP.NET Core. Jde o relativně novou a perspektivní technologii s vysokým výkonem, který dokládá například série benchmarků od společnosti TechEmpower [11]. Platforma nabízí několik možností pro distribuci a následnou instalaci svých aplikací. Význačnou metodou je tzv. *Self-Contained Deployment* (SCD) umožňující distribuci aplikace dohromady s run-time prostředím, na kterém aplikace poběží. Tato funkce usnadňuje následnou instalaci a předchází problémům, které můžou vzniknout při použití jiné verze run-time prostředí. Samotná aplikace může být nasazená, kromě operačního systému Linux, také na Windows nebo MacOS, avšak pro plnou funkčnost rozhraní jsou využity nástroje, které jsou exkluzivní pro operační systém Linux.

4.1.2 Úložiště Snort dat

Ve své aplikaci chci mít možnost zobrazit seznam Snort alertů, které bude zároveň možné rychle rozevřít a zobrazit stránku s detailními pohledem na paket, který tento alert vyvolal. Detailní pohled bude obsahovat přehledně setříděná data z hlavičky paketu, datové pole paketu a tlačítko k jeho stažení ve formátu pcap. K tomu budu potřebovat výstup ze Snortu v unified2 formě obsahující jak alert, tak i příslušný paket. Efektivní způsob, jak vyhledávat konkrétní paket uložený na trvalém úložišti, je za použití databázového serveru. K nahrávaní alertů do databáze mám po vzoru existujících řešení příležitost využít parser Barnyard2.

4.1.3 Zdroj dat ke geolokaci IP adres

Na výběr jsem měl ze dvou možností. Existuje několik internetových Whois databází a geolokačních služeb, ze kterých lze zjistit stát, kterému byla IP adresa přiřazena. Část těchto služeb je placená a druhá část omezuje maximální počet dotazů ze stejné instance. Kromě toho je tento způsob spo-

lehlivý jen v případě nepřetržitého připojení k internetu. Proto jsem se rozhodl využít řešení offline databáze od společnosti MaxMind. Soubor s databází zabírá necelé 4MB a je často aktualizován[17].

4.1.4 Export paketů

Pro funkci exportu paketů bude použit Snort výstup, který zaznamenává podezřelé pakety do souboru pcap. Nutno podotknout, že nejde o úplný paketový záznam, takže velikost souboru by měla být zanedbatelná. Přesto je potřeba dovést filtrovat obsažené pakety a najít v souboru konkrétní paket, který spustil určitý útok. K tomu mám příležitost využít NuGet[18] balíček SharpPcap. Jedná se o adaptér linuxové knihovny libpcap, případně i její varianty pro Windows. K identifikaci jediného paketu potom můžu využít shodu časového razítka vzniku alertu s časovým razítkem paketu obsaženého v pcap souboru. Knihovna dále dovede filtrovat pakety podle různých kritérií. Nabízí se tak vhodná možnost spojit všechny útoky ze stejného zdroje do jednoho souboru k exportu a ucelené analýze.

4.1.5 Wireshark web interface

Wireshark je oblíbený nástroj k analýze exportovaných souborů. Z toho důvodu jsem prozkoumal možnosti, jak napodobit jeho hlavní tabulku pomocí HTML tabulky. Narazil jsem na konzolový program Tshark, pomocí kterého jde vygenerovat prakticky totožné datové sloupce. Tshark se tak stává doporučenou závislostí k nasazení webového rozhraní v praxi.

4.1.6 Webové technologie

K implementaci grafů jsem se rozhodl použít JavaScript knihovnu D3.js. Jako jedna z mála open-source knihoven k vizualizaci dat umožňuje vytvořit jakýkoliv 2D graf s dynamickými prvky, má výbornou dokumentaci a velkou komunitu. JavaScript využiju také pro plynulé asynchronní načítání některých časově náročných komponent prostřednictvím Fetch API. K dlouhodobému zaznamenání uživatelem zvoleného filtru alertů zavedu HTTP Cookie. Takový filtr pak bude moct uživatel aplikovat na více stránek a jejich komponent zároveň.

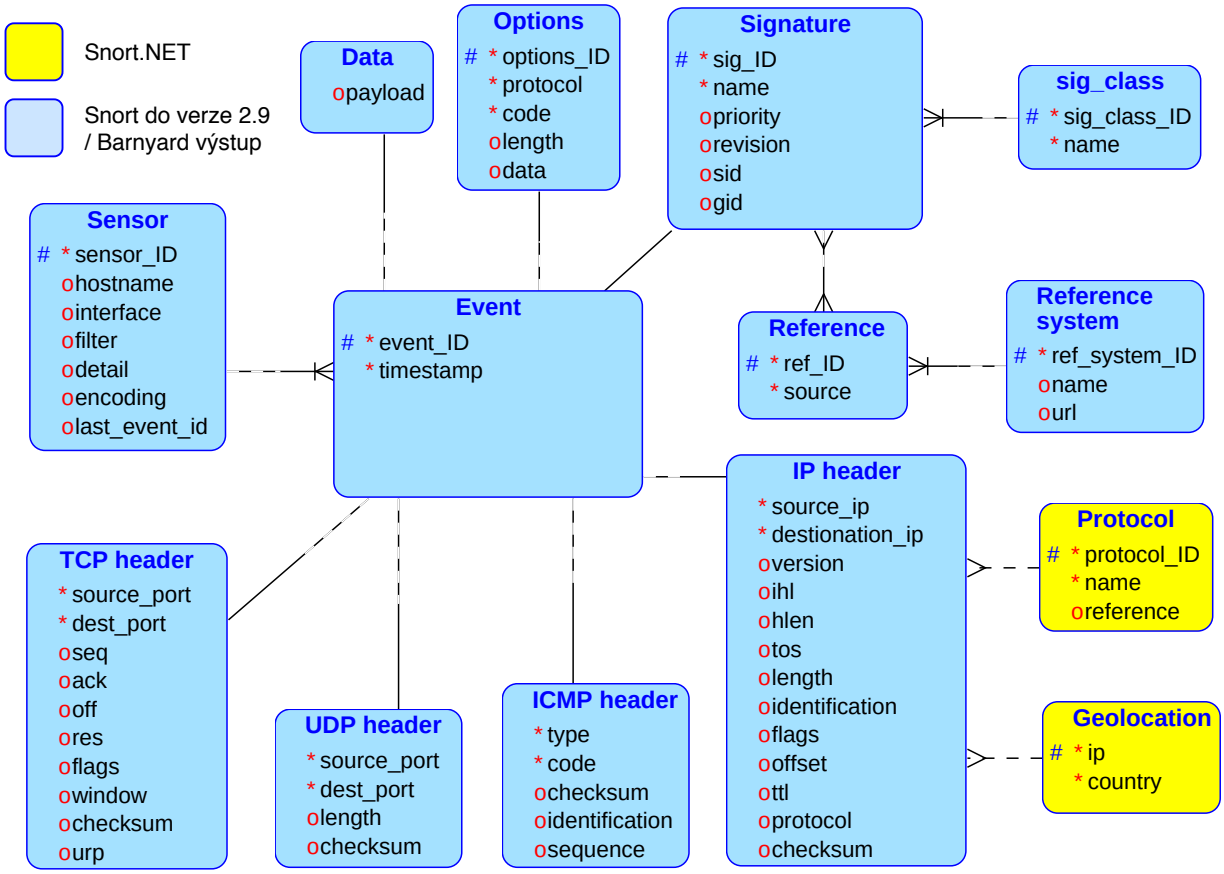
4.2 Datová vrstva

Jako vstupní data jsou použity standardní výstupy z jednoho nebo více Snort senzorů spuštěných v režimu NIDS. Je zapotřebí aktivovat dva výstupní pluginy prostřednictvím konfiguračního souboru `snort.conf`:

1. Snort unified2 alerty – Tyto binární data jsou interpretována v téměř reálném čase pomocí mnou upravené verze nástroje **Barnyard2**. K informacím obsažených v alertu jsou doplněny reference ze souboru *sid-msg.map*, jenž je součástí použitých Snort pravidel. Barnyard tyto data následně ukládá do **MySQL** databáze. Doménový model této databáze je vykreslen na obrázku 5.

[17] https://dev.maxmind.com/geoip/geoip2/geolite2/
[18] NuGet je balíčkovací systém pro volně šiřitelné knihovny v rámci platformy .NET

2. Snort pcap výstup – surový záznam podezřelých paketů je uložen do souboru ve formátu pcap. Aplikace prezentuje tyto data v rámci webového rozhraní s pomocí .NET knihovny SharpPcap. Uživatel má také možnost tyto data stáhnout za účelem další analýzy.



Obrázek 5: Entitně vztahový model datové domény

4.3 Aplikační vrstva

Pro objektově relační zobrazení jsem použil následující návrhové vzory:

- Dependent mapping a Identity field pro tabulku Event
- Table module a Table data gateway pro ostatní tabulky

Použitý webový framework ASP.NET Core – Razor pages vychází z návrhového vzoru Model–view–viewmodel (MVVM).

Při tvorbě webových stránek jsem využil technologií:

- **AJAX** – pro interaktivní prvky a asynchronní načítání

- **D3.js** – javascript knihovna k vykreslení několika grafů za pomocí SVG elementů.
- **bootstrap** – pro kaskádové styly elementů

Web obsahuje 8 hlavních stránek:

- */Index* – Upozorňuje na nejnovější hrozby, které ilustruje pomocí časového histogramu. Rozsah zobrazení může uživatel přepnout na den, měsíc nebo celý rok. Podle vybraného časového úseku se také vykreslí koláčové grafy procentuálního rozložení zdrojových i cílových IP adres. Následuje rozložení typu útoků na časové ose a tabulka popisující posledních 10 vzniklých hrozeb.
- */Events* – Tabulka všech útoků, kterou lze setřídit a také filtrovat podle jednoho nebo několika parametrů zároveň.
  - */Detail* – Stránka s detailními informacemi k útoku. Přehledný přepis celého paketu s možností jeho stažení ve formát pcap. Dokládá popis použitých protokolů, typ útoku s odkazy k externím pramenům a informacím z Whois databáze[19].
- */Protocols* – Zobrazuje vícestupňový interaktivní graf rozložení útoků podle zdrojového nebo cílového protokolu transportní vrstvy (Typicky TCP/UDP/ICMP).
- */Endpoints* – Grafy procentuálního rozložení útoků podle zdrojové a cílové IP adresy
- */Signatures* – Graf procentuálního rozložení útoků podle jeho typu.
- */Origin* – Obsahuje mapu světa, která barevně zvýrazňuje státy podle četnosti útoků z nich pocházejících. Pro tuto stránku přikládám sekvenční diagram, jenž zachycuje komunikaci serveru s webovým prohlížečem a sled procesů v reakci na tento webový dotaz. Viz obrázek 6.
- */About* – Informace k projektu a jeho funkcím, kontakt na autor a místo k nahlašování chyb.

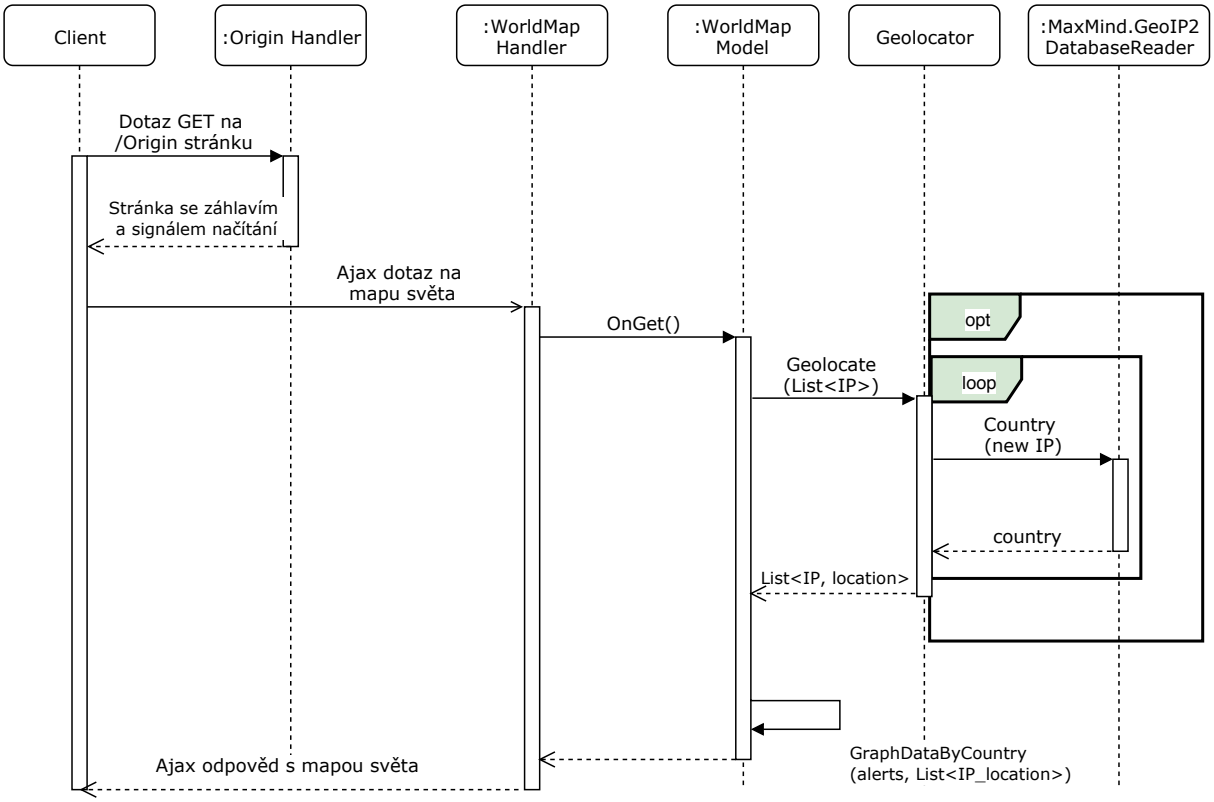
Zdrojový kód každé stránky je obsažen ve dvou souborech elektronické přílohy podle následujícího vzoru:

1. Pohled stránky – `/zdroj/Pages/<název stránky>.cshtml`
2. Model stránky – `/zdroj/Pages/<název stránky>.cshtml.cs`

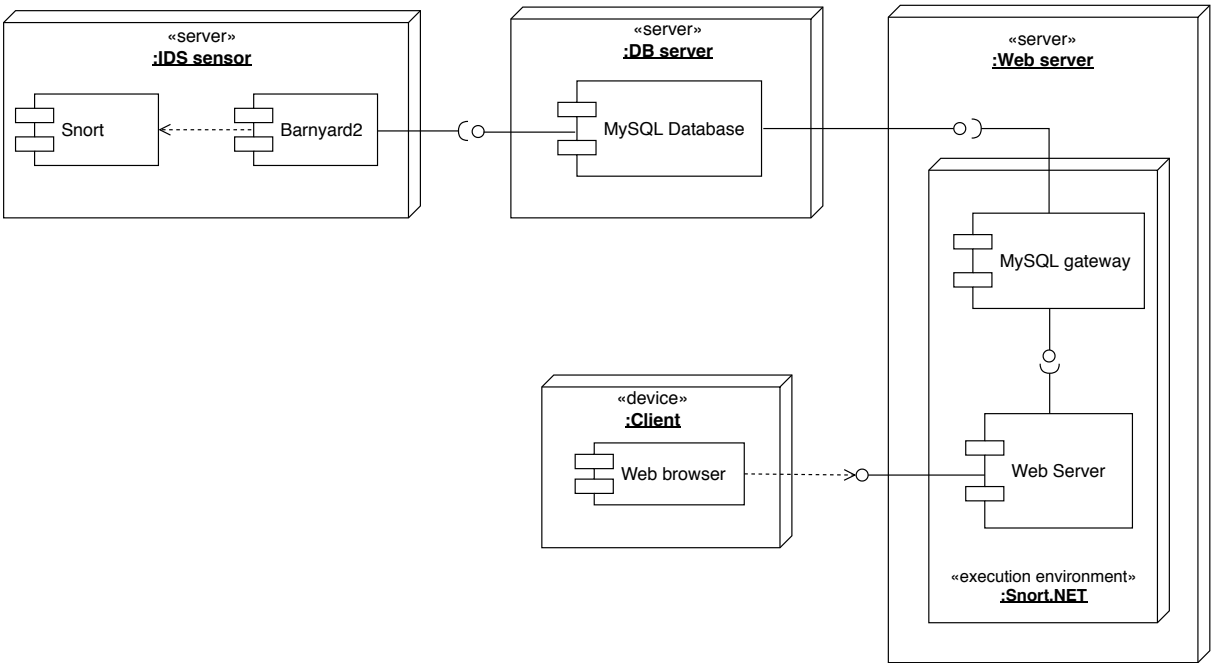
4.4 Architektura systému

Systém lze rozdělit do tří bloků, tj. serverů, které odpovídají diagramu na obrázku 7. Pro účely testování byly tyto servery umístěny v rámci jednoho operačního systému, což ale není podmínkou.

[19] webová služba https://www.ripe.net/



Obrázek 6: Sekvenční diagram požadavku na stránku zobrazující geografická data



Obrázek 7: Diagram nasazení a vztahů jednotlivých komponent systému

- IDS senzor – Obsahuje Snort nastavený do režimu NIDS a nástroj Barnyard, který útoky ukládá do databáze. Senzor umístíme na hranici chráněné sítě s vnější sítí. V případě potřeby může být senzorů více na různých místech sítě.
- databázový server - Úložiště poskytující rozhraní pro rychlé vyhledávání dat.
- web server – Webová aplikace Snort.NET, která se připojuje k databázi skrze gateway komponentu.
- klient – Aplikace plně podporuje webové prohlížeče Google Chrome, Firefox, Opera. Dále také mobilní varianty Dolphin a Google Chrome, určené pro Android zařízení. Kód stránek byl vyladěn a otestován pro plnou funkčnost a souhlasné rozložení prvků na výše zmíněných prohlížečích.

4.5 Testování a hardwarové požadavky

V této sekci budu prezentovat výsledky, kterých jsem dosáhl po optimalizaci aplikace s využitím moderních profilovacích nástrojů. Největší pozornost jsem věnoval rychlosti zpracování webových dotazů a potřebné operační paměti.

Provedl jsem testy nefunkčních požadavků systému a jeho komponent na dvou zařízeních:

1. Windows 10 – Intel Core i5-3570k, (4 × 3.4GHz), 8GB RAM
2. Virtuální Debian 9.8 – 1 × 3.4GHz, 1GB RAM

V tabulce 2 je vidět průměrná doba, kterou server potřebuje ke zpracování požadavků na jednotlivé stránky [20]. V rámci testování škálovatelnosti jsou výsledky rozděleny podle celkového počtu Snort alertů uložených v databázi. Množství současně připojených klientů na webový server by ve standardním případě měl být velmi nízký. Takže v souladu s předpokládaným vytížením serveru jsou provedené testy a jejich výsledky dostačující.

Aplikace byla rovněž vyzkoušena na routeru typu APU.2C4[21]. Z kombinace použitých testovacích zařízení a výsledků těchto testů dále také odvozuji doporučené hardwarové požadavky v tabulce 3.

4.6 Instalace

Zde uvádím obecný postup k instalaci kompletního systému zahrnující všechny jeho komponenty a jejich závislosti. Všechny části mohou být nasazeny v rámci jednoho operačního systému nebo rozloženy podle diagramu na obrázku 7.

V příloze A.3 dále přikládám podrobný návod k instalaci pro operační systém Debian v9.8 (architektura amd64).

[20] Analýza nebere v potaz zpoždění způsobené přenosem a zpracováním na straně klienta
[21] https://pcengines.ch/apu2c4.htm

Tabulka 2: Doba zpracování dotazu v závislosti na systému a počtu databázových záznamů

	Windows 10		Debian VM	
	10 000	100 000	10 000	100 000
Stránka	Průměrná doba zpracování [ms]			
Index	33	257	94	737
Events	24	27	29	46
Detail	515	510	578	540
Protocols	15	16	6	8
Endpoints	12	41	8	78
Signatures	11	18	4	18
Origin	30	46	26	85
Inicializace	1057	2639	2448	3899

Tabulka 3: Doporučené systémové požadavky

Operační systém	Linux-x64/Linux-arm/Windows/OSX-x64/Tizen
CPU	1GHz+
Paměť	1GB RAM

1. **Snort** – Provedeme standardní instalaci Snortu pomocí instalačního balíčku nebo zdroje k dispozici na oficiálních stránkách Snortu[22]. V případě instalace ze zdroje je zapotřebí také připravit všechny nezbytné závislosti. Konkrétně jde o knihovny autotools, daq, g++, hwloc, LuaJIT, OpenSSL, pcap, pcre, pkgconfig a zlib. [12]

   V dalším kroku získáme sbírku pravidel pro Snort. Doporučuji využít oficiální distribuci pravidel pro Snort od skupiny Talos[23]. Pro snadnou instalaci a správu pravidel můžeme využít nástroj Pulledpork[24].

   Následuje základní konfigurace souboru *snort.conf*. Základem je adekvátní definice proměnných `HOME_NET` a `EXTERNAL_NET`. Dále nadefinujeme adresář s pravidly a aktivujeme jednotlivé soubory s pravidly podle jejich kategorií. Zásadní je také správné nastavení dvou výstupních pluginů následujícím způsobem:

```
output unified2: filename merged.log, limit 128
output log_tcpdump: tcpdump.log
```

2. **Barnyard2 a MySQL** – Stáhneme vyladěnou verzi tohoto nástroje, která obsahuje aktualizovaný script k vytvoření MySQL databáze[25]. Připravíme si funkční MySQL server i klient, spustíme zmiňovaný script a vytvoříme nového uživatele pro připojení k této databázi. Před instalací

[22] https://www.snort.org
[23] https://www.snort.org/downloads#rules
[24] https://github.com/shirkdog/pulledpork
[25] https://github.com/fujavica/barnyard2

Barnyardu ho nakonfigurujeme s přepínači `--with-mysql` a `--enable-ipv6`. Podle potřeby ještě zadáme cestu k MySQL knihovnám pomocí přepínače `--with-mysql-libraries=<absolutní cesta>`.

Konfigurační soubor `barnyard.conf` najdeme uvnitř staženého archivu ve složce `etc/`. Pro korektní inicializaci programu a databáze zadáme správnou cestu ke čtyřem souborům [26], které popisují statické informace použitých pravidel z kroku 1. V poslední části konfigurace aktivujeme výstup do MySQL databáze a vyplníme údaje k jejímu připojení.

Abychom mohli skrze webové rozhraní reagovat na vzniklé hrozby ihned po jejich detekci Snortem, spustíme Barnyard v režimu neustálého běhu jako daemon. Při spuštění mu zadáme cestu k Snort unified2 souborům pomocí argumentů přepínačů `-d` a `-f`. K tomu, aby program neprocházel zpracované alerty znova při každém spuštění, vytvoříme záložku, respektive prázdný soubor a zadáme jeho cestu jako argument přepínače `-w`.

3. **Snort.NET** – Jestliže hostujeme webový server na linuxové platformě, jako první nainstalujeme nástroje pro práci s paketovým záznamem souborů pcap: tcpdump a tshark. Samotná webové aplikace je k dispozici na githubu jako SCD distribuce[27]. V rozbaleném archivu najdeme soubor `appsettings.json`, kde vyplníme údaje pro připojení k MySQL serveru a cestu ke Snort logům.

   Aplikace disponuje vlastním multiplatformním web serverem Kestrel. Ve výchozím nastavení bude rozhraní poslouchat na portu 5000. Změnit ho můžeme pomocí následujícího startovacího argumentu:

```
./razor --urls=http://localhost:5001/
```

[26] `reference.config`, `classification.config`, `gen-msg.map`, `sid-msg.map`
[27] https://github.com/fujavica/Snort.NET/releases

5 Srovnání

Pro srovnání se svou implementací jsem si vybral dále rozhraní BASE a Snorby, jež mají podobný design a pestrý výběr aspektů pro porovnání. Všechny 3 rozhraní byly spuštěny v totožném virtuálním prostředí (Debian 9.8 – 1 × 3.4GHz, 1GB RAM) a se stejnými vstupními daty ze Snortu. Celkově bylo aplikacím poskytnuto 38 898 záznamů útoků, z toho 425 rozdílných typů útoků spuštěných z 2613 různých zdrojových IP adres.

5.1 Úvodní stránka

Hlavní stránka pro rozhraní jakéhokoliv bezpečnostního systému by měla být schopná upozornit uživatele na nejnovější události v systému bez zbytečné prodlevy. Testované aplikace tohle zvládají dobře a k načtení nových hrozeb stačí obnovení stránky. Dále se budu věnovat specificky jednotlivým rozhraním.

- BASE – Stránka nalevo sumarizuje celkový objem různých dat (počet senzorů, alertů, IP adres a portů). Napravo je graficky znázorněný poměr protokolů (TCP, UDP, ICMP). Jde o relativně stručnou úvodní stránku, což je samo o sobě v pořádku. Z těchto informací však uživatel nezjistí nejnovější hrozby a musí si je explicitně zobrazit pomocí některého z odkazů v záhlaví této stránky. Snímek stránky je zachycen na obrázku 2a.

- Snorby – Srdcem úvodní stránky je počet alertů podle úrovně závažnosti (1 – 3), dále obsahuje 3 druhy koláčových grafů (k porovnání četnosti signatur, zdrojových a cílových IP adres) a 3 druhy spojnicových grafů (k porovnání četnosti zasažených senzorů, použitých protokolů a stupňů závažnosti na časové ose). Mezi grafy lze svižně přepínat a také měnit časový úsek ze kterého berou data. Stránka je zachycena na obrázku 3.

- Snort.NET – Pro svou úvodní stránku jsem se inspiroval hlavně ze Snorby a postupně jsem přidal další komponenty. Jako první je zde histogram zobrazující počet alertů ve vybraném časovém intervalu. K přepínání intervalu slouží ovládací komponenta s tlačítky znázorňující čtyři směry. Šipkami doprava/doleva se úsek posouvá o jeden den, rok nebo měsíc dopředu/dozadu a šipkami dolu/nahoru lze přepnout časový rozsah na den, rok nebo měsíc. Následují dva koláčové grafy popisující četnost zdrojových a cílových IP adres ve vybraném časovém intervalu. Jednotlivé části těchto grafů a jejich popisky jsou propojené se seznamem alertů a po kliknutí se aplikuje filtr na danou IP adresu. Dále je zde tabulka s 10 nejnovějšími událostmi, které Snort zaznamenal s barevným odlišením řádků podle závažnosti. Poslední je posuvná časová osa, tentokrát rozdělená do více řádků, kde jeden řádek odpovídá jedné signatuře, tj. typu útoku. Úkolem komponenty je ukázat četnost jednotlivých signatur ve vybraném časovém horizontu.

  Snort.NET přináší několik interaktivních vizuálních prvků, které by měli být užitečné k bezpečnostní analýze v libovolném časovém horizontu. Prvky jsou v porovnání s konkurencí hodně

velké a stránka tak vyžaduje dost posunování. Na druhou stranu, jsou ale lépe propojené se zbytkem webu. Snímek stránky je zachycen v příloze B na obrázcích 10 a 11.

5.2 Detail útoku

K pokročilé analýze jakéhokoliv alertu, slouží komponenta, která agreguje co nejvíc užitečných informací k jeho upřesnění. Typicky obsahuje paketu rozložený do jednotlivých vrstev TCP/IP modelu a jednotlivých datových polí těchto vrstev.

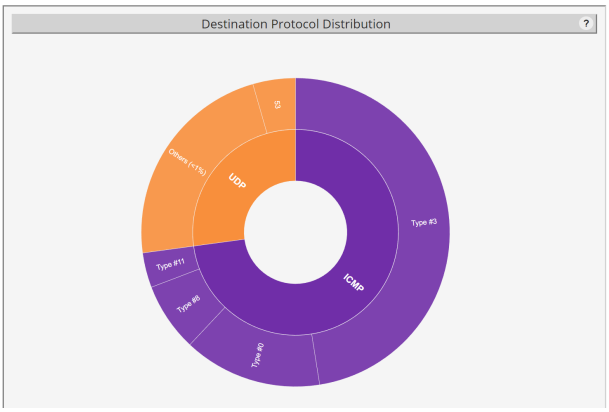
- BASE – Obsahuje signaturu útoku s externími odkazy k bližšímu popisu, rozložený paket s odkazy k použitým protokolům, data z Whois registrů pro zdrojovou i cílovou IP adrese, tlačítko ke stažení paketu ve formátu pcap.
- Snorby – přidává možnost zobrazit přesnou definici Snort pravidla, spjatého s daným útokem a řadu způsobů, jak s konkrétním alertem dále naložit. Analytik má za úkol alert zařadit do jedné z 8 nadefinovaných kategorií, případně k němu přidat svou poznámku. Detail může být také rozšířen o integraci nástroje OpenFPC pro úplný paketový záznam.
- Snort.NET – Má všechno, co BASE a na konci tabulku, která shrnuje všechny útoky, tj. pakety ze stejného zdroje. Sled paketů je možné také exportovat do tcpdump formy. Poslední tabulka má ambici z části nahradit některé jiné paketové analyzátory (tcpdump, wireshark) a je exkluzivní pro rozhraní Snort.NET.

5.3 Graf rozdělení použitých protokolů

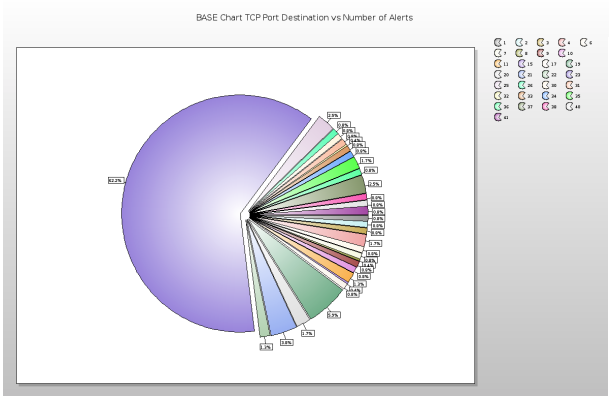
Na tomto místě se budu snažit porovnat vzhled a možnosti grafů zachycujících statistická data, na konkrétním příkladu. Příkladný graf má zachytit četnost použití komunikačních protokolu, zejména na úrovní transportní vrstvy (nejčastěji jde o TCP, UDP a ICMP). Tyto protokoly jde dále rozdělit podle dalších atributů jim vlastních (port u TCP a UDP, typ u ICMP). V praxi to znamená, že analytik má možnost identifikovat, které konkrétní poskytované služby jeho systému jsou zneužívané k útoku nebo i opačně, které externě využívané služby se nechovají korektně.

- BASE – Dovede vykreslit řadu grafů v třech formách (sloupcový, spojnicový, koláčový) včetně těch, co zachycují zdrojové a cílových porty zvlášť pro TCP a UDP protokol. V testových podmínkách dotazy trvaly relativně dlouho a velké množství portů bylo zachyceno v příliš nahuštěné formě. Rozhraní sice umožňuje regulovat celkový počet vykreslených portů, ty jsou ale vybrány podle jejich číselné hodnoty namísto jejich celkové četnosti. Ukazuje se, že BASE v této oblasti není dobře koncipovaný na velké množství dat a ilustrace vyžadují vstup spíše z krátkých časových intervalů.
- Snorby – Na své úvodní přehledové stránce má spojnicový graf porovnávající četnost použití TCP, UDP a ICMP protokolů. Chybí zde však četnost konkrétních portů na rozdíl od ostatních aplikací.

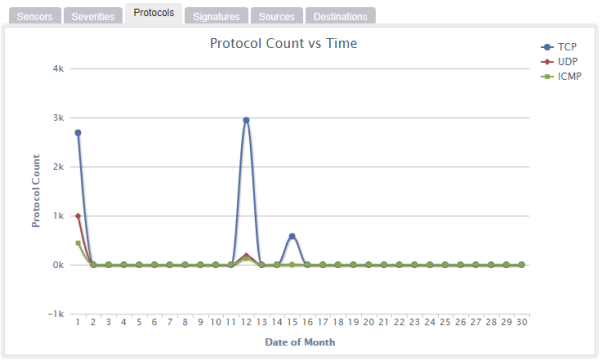
- Snort.NET – Nabízí dvouvrstvý interaktivní koláčový graf ve dvou provedeních pro cílové a zdrojové služby (porty). Vnitřní vrstva porovnává četnost mezi transportními protokoly (TCP, UDP, ICMP) a vnější vrstva porovnává jejich jednotlivé porty/typy. Komponenta je propojená s filtrem alertů, tj. při kliknutí na libovolný port se otevře seznam útoků spojených s tímto portem. Na vstupní data všech grafů jdou uplatnit filtry alertů, což jejich použití posouvá o krok dále oproti konkurenci.



(a) Snort.NET



(b) BASE



(c) Snorby

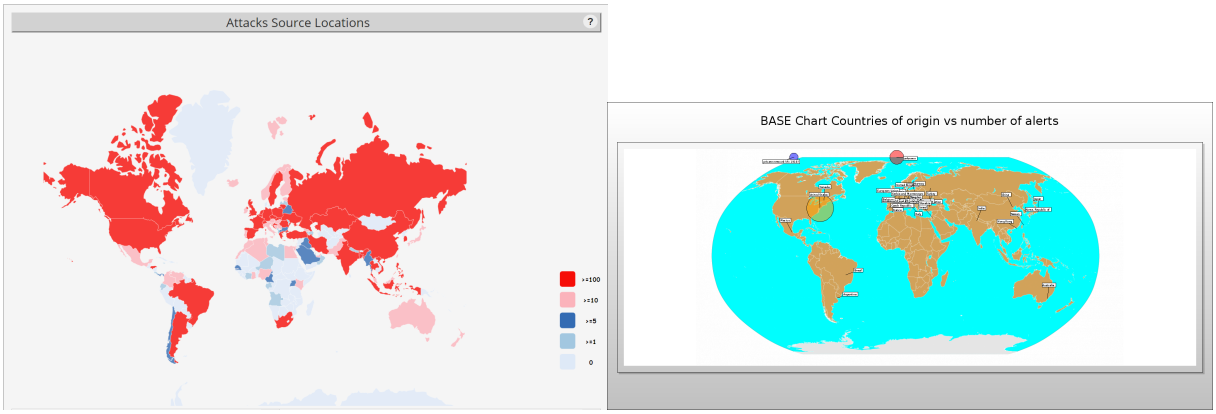
Obrázek 8: Graf rozdělení použitých protokolů

5.4 Mapa distribuce původu útoků

Jak bylo předesláno v kapitole 3.5, všechny zmíněné řešení obsahují funkce ke geolokaci IP adres. Existuje více možností, jak tyto informace získat. Vybraný způsob nakonec může ovlivnit to, jak moc jsou data aktuální a spolehlivá. Jejich pravdivost se zároveň snižuje s tím, na jak přesné umístění se dotazujeme. Některé služby ke geolokaci IP adres uvádějí 95% – 99% přesnost pro umístění na úrovni státu, 55% – 80% přesnost pro region původu a pouze 50% – 75% přesnost pro město původu [13]. V důsledku toho, tak aplikace nejčastěji uvádějí pouze stát původu jako orientační ale spolehlivou

informaci. S nasbíraných záznamů útoků se nabízí možnost vykreslit světovou mapu se zvýrazněnými státy podle počtu evidovaných útoků.

- BASE – Umožňuje vykreslit mapu světa, kde jsou označeny státy, ze kterých byl veden alespoň jeden útok pomocí štítků. Množství spuštěných útoků z jednotlivých států ale není nijak vyznačeno a štítky se navzájem překrývají. V testovaném případě, který obsahoval 2613 různých zdrojů byl problém se štítky obzvlášť patrný. Doba zpracování stránky trvala okolo 100 sekund při každém novém požadavku. Celkově tedy mapa obsahuje málo relevantních informací a její implementaci považuju jako nevyhovující.
- Snorby – Nedisponuje takovou komponentou.
- Snort.NET – Stránka */Origin* obsahuje mapu světa s barevným zvýrazněním státu podle počtu zachycených útoku. Čím je počet útoků vyšší, tím se zabarvení států mění z bledě modré až na sytě červenou (barva se mění ve skocích na rozmezí 1, 5, 10, a 100 útoků). Přesný počet útoků z jednoho státu lze zjistit po označením státu kurzorem nebo v tabulce pod mapou. Ke geolokaci je využívána aktualizovaná offline databáze, která pracuje svižně. Při prvním pokusu o zobrazení, u kterého bylo zapotřebí lokalizovat 2613 IP adres, tento proces zabral 25 sekund. Při opětovném zobrazení stránky byla již data v mezipaměti a zpoždění zanedbatelné.



(a) Snort.NET (b) BASE

Obrázek 9: Mapa distribuce původu útoků

6 Závěr

Cílem práce bylo navrhnout nové webové rozhraní pro IDS Snort. Detailně jsem prozkoumal několik stávajících webových rozhraní, konkrétně šlo o rozhraní SnortAlog, BASE, Snorby a Sguil. Následně jsem porovnal jejich funkce a v některých případech jsem rozpoznal prostor pro jejich vylepšení. Na základě návrhu jsem implementoval webovou aplikaci Snort.NET. Jako platformu k implementaci jsem zvolil ASP.NET Core v aktuálně stabilní verzi 2.2. Jde o aktivně vyvíjený open-source projekt se skvělými vývojovými nástroji a s podporou mnoha operačních systémů. Součástí aplikace je zabudovaný webový server Kestrel a ve zvolené variantě pro distribuci (SCD) i samotné run-time prostředí, takže před spuštěním je potřeba jen upravit konfigurační soubor podle svých potřeb. V příloze přikládám zdrojový kód aplikace a distribuční balíček pro 64-bitové linuxové systémy. Kód a balíčky pro ostatní platformy jsou dostupné na GitHub stránce projektu[28]. Aplikace byla úspěšně zprovozněna a otestována ve virtuálním prostředí a posléze také na routeru typu APU.2C4. Ze zátěžových testů nevyplynuly žádné překážky pro datové vzorky o velikosti desítky až stovky tisíc Snort záznamů. Web je tvořen kolekcí stránek obsahující několik grafických komponent k vizualizaci statistických dat, filtrovatelný seznam hrozeb a jejich detailní výpis s možností stáhnout zachycený paket ve formátu pcap.

V kapitole 5 jsem svou aplikaci zpětně srovnal s její podobnými. S využitím JavaScriptu a SVG technologií je moje webové rozhraní schopné nabídnout lepší vizualizaci dat a propojení webu než jiné alternativy. Dále umí pracovat s IPv6 adresami, zatímco většina starších implementací na ně pořád není připravena. Do detailního výpisu útoku jsem navíc integroval výstup z nástroje Wireshark, který popisuje sled útoků ze stejného zdroje.

Možnost pro další vývoj aplikace vidím v lepším propojení se Snortem, tak aby analytik mohl manipulovat se Snort pravidly přímo z uživatelsky přívětivého webové rozhraní. Dále chybí možnost zabezpečeného přihlášení a následně uživatelského přizpůsobení barev nebo uspořádání komponent úvodní stránky, což některé komerční řešení nabízí. Aplikace je závislá na programu Barnyard, jehož funkce by mohla být integrována v rámci této aplikace, což by usnadnilo instalaci a manipulaci se Snort pravidly. V případě zájmu se prostor nabízí také v rámci lokalizace rozhraní do více jazyků.

[28] https://github.com/fujavica/Snort.NET

Literatura

[1] Knud Lasse Lueth. State of the iot 2018: [online], 2018. [cit. 2019-03-16]. Dostupné z: https://iot-analytics.com/state-of-the-iot-update-q1-q2-2018-number-of-iot-devices-now-7b/.

[2] Rafeeq Ur Rehman. *Intrusion Detection Systems with Snort: Advanced IDS Techniques Using Snort, Apache, MySQL, PHP, and ACID*. Prentice Hall, 2003. ISBN 0-13-140733-3.

[3] Peter Mell and Karen Scarfone. Guide to intrusion detection and prevention systems (idps)), 2007. [cit. 2019-03-16]. DOI: 10.6028/NIST.SP.800-94. Dostupné z: https://csrc.nist.gov/publications/detail/sp/800-94/final.

[4] A. L. P. d. França, R. P. Jasinski, V. A. Pedroni, and A. O. Santin. Moving network protection from software to hardware: An energy efficiency analysis. In *2014 IEEE Computer Society Annual Symposium on VLSI*, pages 456–461, 2014. [cit. 2019-03-16]. DOI: 10.1109/ISVLSI.2014.89. Dostupné z: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6903406.

[5] Jay Beale, Andrew R. Baker, and Joel Esler. *Snort IDS and IPS Toolkit (Jay Beale's Open Source Security)*. Syngress, 2007. ISBN 1-59749-099-7.

[6] Doug Dineley. The greatest open source software of all time [online], 2009. [cit. 2019-03-16]. Dostupné z: https://www.infoworld.com/article/2631146/the-greatest-open-source-software-of-all-time.html.

[7] The Snort Release Team. Snort 3 beta available now [online], 2018. [cit. 2019-03-16]. Dostupné z: https://blog.snort.org/2018/08/snort-3-beta-available-now.html.

[8] Sébastien Damaye. Suricata-vs-snort [online], 2016. [cit. 2019-03-16]. Dostupné z: https://www.aldeid.com/wiki/Suricata-vs-snort.

[9] S Chakrabarti, Mohuya Chakraborty, and Indraneel Mukhopadhyay. Study of snort-based ids. In *Proceedings of the International Conference and Workshop on Emerging Trends in Technology - ICWET '10*, pages 43–47. ACM, 2010. [cit. 2019-03-16]. DOI: 10.1145/1741906.1741914. Dostupné z: https://dl.acm.org/citation.cfm?id=1741914.

[10] Richard Bejtlich. *The Tao of Network Security Monitoring: Beyond Intrusion Detection*. Addison Wesley, 2004. ISBN 0-321-24677-2.

[11] TechEmpower. Web framework benchmarks [online], 2018. [cit. 2019-03-16]. Dostupné z: https://www.techempower.com/benchmarks/.

[12] Martin Roesch & The Snort Team. Snort manual [online], 2018. [cit. 2019-03-16]. Dostupné z: https://snort.org/downloads/snortplus/snort_manual.html.

[13] iplocation.net. How accurate is ip-based geolocation lookup, 2018. [cit. 2019-03-16]. Dostupné z: https://www.iplocation.net/geolocation-accuracy.

A Návod k instalaci Snort.NET pro Debian 9.8

Anglická verze tohoto návodu je k dispozici také v GitHub repozitáři projektu[29].

A.1 Instalace Snortu ze zdroje

1. Ujistíme se, že máme aktualizovanou databázi balíčků.

```
apt-get update && apt-get upgrade
```

2. Provedeme instalaci potřebných knihoven.

```
apt-get install -y build-essential gcc make libpcre3-dev zlib1g-dev libluajit-5.1-dev
    libpcap-dev openssl libssl-dev libnghttp2-dev libdumbnet-dev bison flex libdnet
```

3. Stáhneme a nainstalujeme Acquisition library (DAQ).

```
wget https://www.snort.org/downloads/snort/daq-2.0.6.tar.gz
tar xvzf daq-2.0.6.tar.gz
cd daq-2.0.6
./configure && make && sudo make install
```

4. Stáhneme a nainstalujeme Snort.

```
wget https://www.snort.org/downloads/snort/snort-2.9.12.tar.gz
tar xvzf snort-2.9.12.tar.gz
cd snort-2.9.12
./configure --enable-sourcefire && make && sudo make install
```

5. Vytvoříme uživatele a adresářovou strukturu pro Snort.

```
sudo groupadd snort
sudo useradd snort -r -s /sbin/nologin -c SNORT_IDS -g snort
sudo mkdir /etc/snort
sudo mkdir /var/log/snort
sudo mkdir /usr/local/lib/snort_dynamicrules
sudo chmod -R 5775 /etc/snort
sudo chmod -R 5775 /var/log/snort
sudo chmod -R 5775 /usr/local/lib/snort_dynamicrules
sudo chown -R snort:snort /etc/snort
sudo chown -R snort:snort /var/log/snort
sudo chown -R snort:snort /usr/local/lib/snort_dynamicrules
cp etc/* /etc/snort
```

[29] https://github.com/fujavica/Snort.NET/wiki

6. Obdržíme Talos pravidla [30]. Na výběr máme ze 3 úrovní těchto pravidel. Doporučuji využít pravidel 2. úrovně, jež jsou pro registrované uživatele zdarma. Balík s pravidly potom rozbalíme třeba do `/etc/snort`.

7. Nakonfigurujeme */etc/snort/snort.conf*

   - Určíme adresu segmentu chráněné sítě.

     ```
     ipvar HOME_NET server_public_IP/32
     ```

   - Zadáme cestu k pravidlům.

     ```
     var RULE_PATH /etc/snort/rules
     var SO_RULE_PATH /etc/snort/so_rules
     var PREPROC_RULE_PATH /etc/snort/preproc_rules
     ```

   - Aktivujeme výstupy ve formátu unified2 a tcpdump

     ```
     output unified2: filename snort.log, limit 128
     output log_tcpdump: tcpdump.log
     ```

8. Spustíme Snort v NIDS režimu

   ```
   sudo snort -c /etc/snort/snort.conf -u snort -g snort -D
   ```

A.2 Instalace programu Barnyard2 a MySQL

1. Provedeme instalaci potřebných knihoven.

   ```
   apt-get install -y mysql-server mysql-client autoconf libtool libmariadbclient-dev
       default-libmysqlclient-dev libgmp-dev
   ```

[30] https://www.snort.org/downloads#rules

2. Stáhneme a nainstalujeme Barnyard2:

```
wget https://github.com/fujavica/barnyard2/archive/master.tar.gz -O
    barnyard2-Master.tar.gz
tar zxvf barnyard2-Master.tar.gz
cd barnyard2-master
autoreconf -fvi -I ./m4
sudo ln -s /usr/include/dumbnet.h /usr/include/dnet.h
sudo ldconfig
./configure --enable-ipv6 --with-mysql --with-mysql-libraries=/usr/lib/x86_64-linux-gnu
make
sudo make install
```

3. Vygenerujeme soubor mapující Snort pravidla, který bude potřebný k počátečnímu vyplnění databáze.

```
python tools/gensidmsgmap.py -2 /etc/snort/rules/!(deleted.rules) > /etc/snort/sid-msg.map
python tools/gensidmsgmap.py -2 /etc/snort/so_rules/ >> /etc/snort/sid-msg.map
```

4. Připravíme MySQL tabulky a uživatele.

```
sudo mysql
create database snort;
USE snort;
source schemas/create_mysql
CREATE USER 'snortdb'@'localhost' IDENTIFIED BY 'YOUR_DB_PASSWORD';
grant create, insert, select, delete, update on snort.* to 'snortdb'@'localhost';
```

5. Připravíme soubory pro Barnyard.

```
sudo cp etc/barnyard2.conf /etc/snort/
sudo mkdir /var/log/barnyard2
sudo chown snort /var/log/barnyard2
sudo touch /var/log/snort/barnyard2.waldo
sudo chown snort /var/log/snort/barnyard2.waldo
```

6. Nakonfigurujeme soubor: */etc/snort/barnyard2.conf*
   - Nastavíme přesnou cestu k souborům, které popisují použitá pravidla a jejich reference.

     ```
     config reference_file: /etc/snort/etc/reference.config
     config classification_file: /etc/snort/etc/classification.config
     config gen_file: /etc/snort/gen-msg.map
     config sid_file: /etc/snort/sid-msg.map
     ```

   - Vyplníme uživatele a heslo pro připojení k databázi.

     ```
     output database: log, mysql, user=snortdb password=YOUR_DB_PASSWORD dbname=snort
         host=localhost
     ```

7. Pří spouštění zadáme cestu ke Snort logům a k jejich záložce.

```
barnyard2 -c /etc/snort/barnyard2.conf -d /var/log/snort -f merged.log -w
    /var/log/snort/barnyard2.waldo
```

A.3 Instalace webového rozhraní Snort.NET

1. Nainstalujeme nástroje potřebné k analýze paketů ve formátu pcap.

```
sudo apt-get install tshark tcpdump
sudo ln -s /usr/sbin/tcpdump /usr/local/bin/tcpdump
```

2. Stáhneme a rozbalíme aplikační balík v SCD distribuci pro hostitelskou platformu[31].

3. Provedeme konfiguraci souboru: *appsettings.json*.

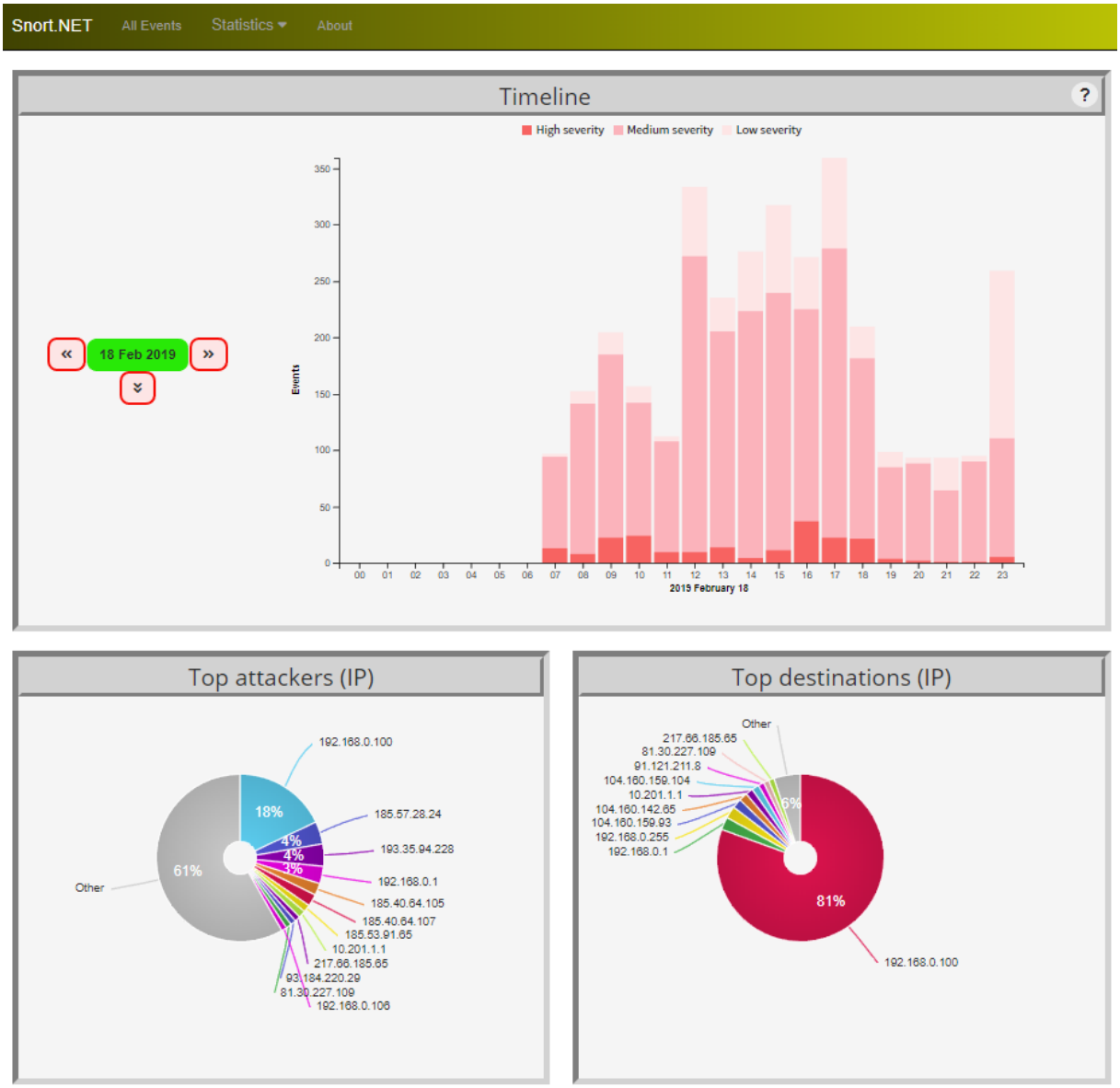
   - Zadáme údaje pro připojení k MySQL databázi.
   - Zadáme cestu k záznamu podezřelých paketů (pcap výstup ze Snortu).

4. Aplikaci můžeme spustit.

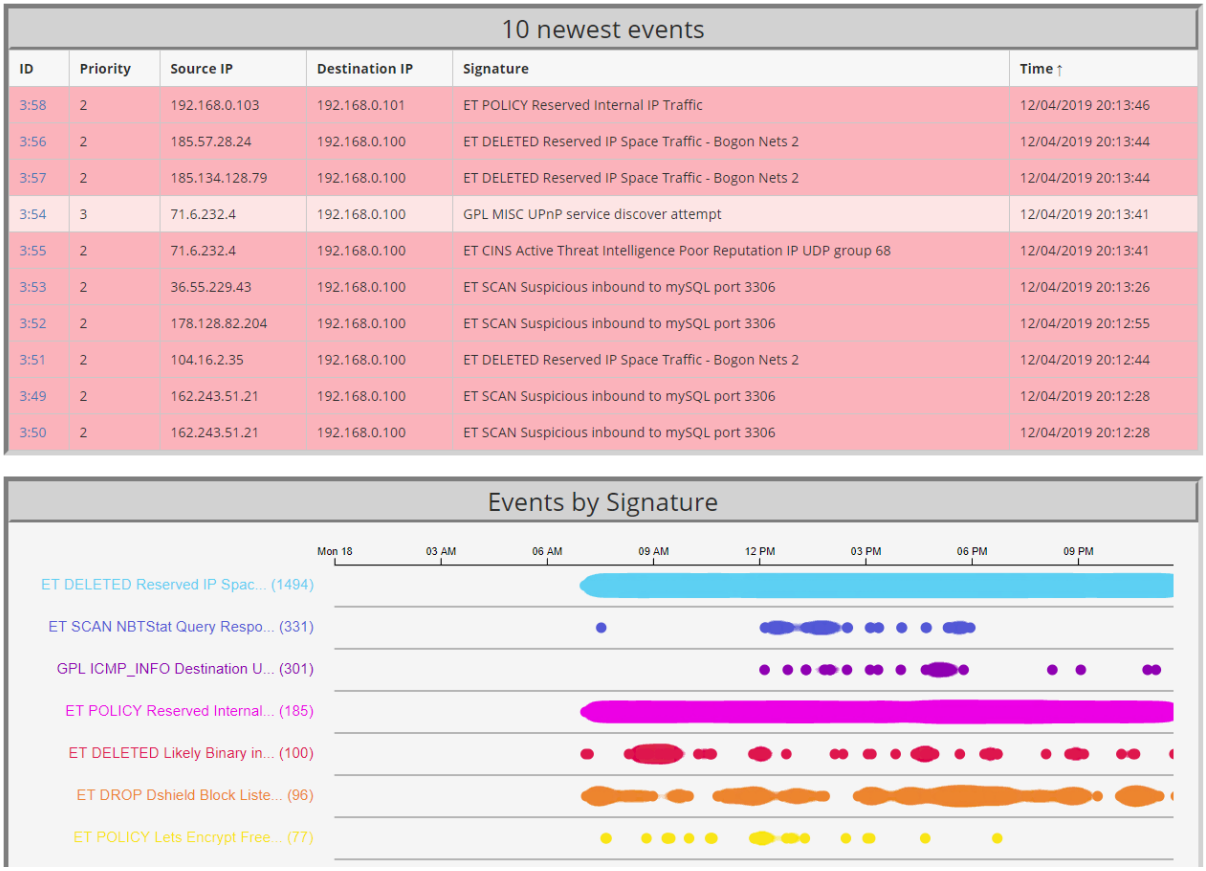
```
./razor --urls=http://localhost:5000/
```

[31] https://github.com/fujavica/Snort.NET/releases/

B Úvodní stránka Snort.NET



Obrázek 10: První část úvodní stránky webového rozhraní Snort.NET. Časové osa v horním části zobrazuje počet útoků ve vybraném časovém intervalu, který lze přepínat pomocí tlačítek nalevo. Šipkami doprava/doleva se úsek posouvá o den, rok nebo měsíc dopředu/dozadu a šipkami dolu/nahoru lze přepnout časový rozsah na den rok nebo měsíc. Následují dva koláčové grafy popisující četnost zdrojových a cílových IP adres ve vybraném časovém intervalu.



Obrázek 11: Druhá část úvodní stránky webového rozhraní Snort.NET. Obsahuje tabulku s 10 nejnovějšími událostmi, které Snort zaznamenal, s barevným odlišením řádků podle závažnosti. Dole je posuvná časová osa rozdělená do více řádků, kde jeden řádek odpovídá jedné signatuře, tj. typu útoku. Úkolem komponenty je ukázat rozložení útoků ve vybraném časovém horizontu v závislosti na jejich signatuře.